University of Alberta

DECOMPOSITION AND COORDINATION OF LARGE-SCALE OPERATIONS OPTIMIZATION

by

Ruoyu Cheng

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

in

Process Control

Department of Chemical and Materials Engineering

Edmonton, Alberta
Fall 2007

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-32937-5
Our file Notre référence
ISBN: 978-0-494-32937-5

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

"There is no substitute for hard work."

- Thomas A. Edison

Abstract

Nowadays, highly integrated manufacturing has resulted in more and more large-scale industrial operations. As one of the most effective strategies to ensure high-level operations in modern industry, large-scale engineering optimization has garnered a great amount of interest from academic scholars and industrial practitioners.

Large-scale optimization problems frequently occur in industrial applications, and many of them naturally present special structure or can be transformed to taking special structure. Some decomposition and coordination methods have the potential to solve these problems at a reasonable speed. This thesis focuses on three classes of large-scale optimization problems: linear programming, quadratic programming, and mixed-integer programming problems. The main contributions include the design of structural complexity analysis for investigating scaling behavior and computational efficiency of decomposition strategies, novel coordination techniques and algorithms to improve the convergence behavior of decomposition and coordination methods, as well as the development of a decentralized optimization framework which embeds the decomposition strategies in a distributed computing environment. The complexity study can provide fundamental guidelines to practical applications of the decomposition and coordination methods.

In this thesis, several case studies imply the viability of the proposed decentralized optimization techniques for real industrial applications. A pulp mill benchmark problem is used to investigate the applicability of the LP / QP decentralized optimization strategies, while a truck allocation problem in the decision support of mining operations is used to study the MILP decentralized optimization strategies.

Acknowledgements

I believe that a journey is easier when you travel together. This thesis is the result of four years of work whereby I have been accompanied and supported by many people. I really appreciate this opportunity to express my sincere gratitude for all of them.

First and foremost, I would like to thank Dr. Fraser Forbes, my PhD supervisor and the department chair, for all his seasoned guidance, great encouragement, deep insight, and patient help during the years that we have been through. He might not realize how much I have learned from him, and I would have been lost without him. I will never forget his advice on explaining things in plain language, from which I have benefited greatly in my research work and later on industrial practice. Besides being an excellent supervisor, Fraser has always been a good friend to his students as well. I am really glad that I have come to know Fraser in my life.

It is difficult to overstate my gratitude to Dr. San Yip, my PhD co-supervisor, who always kept an eye on the progress of my work and was available whenever I needed his advice. He always made me feel welcome to ask him questions, no matter how busy he was, or how silly the questions were. Many good ideas in this thesis came out from the large amount of discussion with San. He is another essential person who helped make this thesis possible.

I would also like to thank the other members of my PhD supervisory committee, who have monitored my work and put a lot of effort in reading and providing me with invaluable comments at earlier stages of this work: Dr. Ming Zuo and Dr. Guohui Lin. I also appreciate the suggestions and encouragement from Dr. Biao Huang and Dr. Sirish Shah, who have always been very generous to offer me any help in my studies.

In my office at the University of Alberta, I was surrounded by knowledgeable and friendly people who helped me every day. I would like to say thanks to Dr. Zhengang Han, Yutong Qi, Fangwei Xu, Adrian Fuxman, and Rumana Sharmin, for establishing such a stimulating work environment and always being so active and helpful in our discussions. In addition, so many thanks to Bob Barton and Jack Gibeau for their excellent computer and network support.

I would like to say thanks to the many scholars from other universities, who provided me with valuable information which allowed me to see a bigger picture of this research area. I must not forget to mention: Dr. James Rawlings, Dr. Frank Doyle, Dr. Ivar Ekland, Dr. Rong Chen, Dr. Mehmet Mercangoz, and Dr. Francesca Fumero.

This research has been supported and funded by various organizations including the Department of Chemical and Materials Engineering at the University of Alberta, NSERC, and Syncrude Canada Limited. I thank them all for their confidence in me. Particularly, I would like to express my gratitude to Dr. James Kresta and Mr. Gary Anthieren from Syncrude, for their inspiring support throughout my research.

Lastly, and most importantly, I wish to express my deepest gratitude to my parents, who formed part of my vision and taught me the good things that really matter in life. I shall be grateful all my life to my wife, Liping, who shares both happiness and sadness with me. It was her love, understanding and patience that supported my hard work during the past few years.

In addition to the above, the thesis is also dedicated to those who have inspired, guided and helped this humble being.

Contents

1 Introduction **1**
1.1 Overview of Large-scale Optimization 2
  1.1.1 Optimization of Large-scale Operations 2
  1.1.2 Solving Large-scale Optimization Problems 4
1.2 Research Scope 7
1.3 Thesis Overview 10

2 Decomposition Strategies for Large-scale Linear Programming **12**
2.1 Background 13
  2.1.1 Linear Programming 13
  2.1.2 Dantzig-Wolfe Decomposition Principle 15
2.2 Column Generation Techniques 18
  2.2.1 Single-column vs. Multi-column Generation 20
  2.2.2 Dantzig-Wolfe Decomposition Algorithms 22
  2.2.3 Discussions 23
2.3 Complexity Study 24
  2.3.1 Theoretical Analysis 25
  2.3.2 Empirical Studies 28
2.4 Chapter Summary 37

3 Decomposition Strategies for Large-Scale Quadratic Programming **39**

3.1 Background 40
  3.1.1 Decomposition Strategies for Nonlinear Programming 40
  3.1.2 Extension of Dantzig-Wolfe Decomposition 41
  3.1.3 Discussions 44
  3.1.4 Price-driven Coordination Methods 45
3.2 An Efficient Price Update Scheme 46
  3.2.1 Newton's Method 46
  3.2.2 Jacobian Evaluation 47
  3.2.3 Step Size Determination 49
  3.2.4 Algorithmic Statement 50
  3.2.5 Discussions 51
3.3 Economic Interpretation of Coordination Mechanism 53
3.4 Complexity Study 55
  3.4.1 Theoretical Analysis 56
  3.4.2 Empirical Studies 60
3.5 Chapter Summary 77

4 Coordinated, Decentralized MPC 79
4.1 Background 80
  4.1.1 MPC Target Calculation 80
  4.1.2 Plant-wide MPC 83
  4.1.3 Coordinated MPC 84
4.2 Coordination of Decentralized MPC 86
  4.2.1 Coordination through a Coordinator 86
  4.2.2 Identification of Linking Constraints 86
4.3 Dantzig-Wolfe Decomposition and Plant-wide MPC 90
  4.3.1 Illustrative Case Studies 90
4.4 Price-driven Coordination and Plant-wide MPC 98
  4.4.1 A Pulp Mill Benchmark Process 98
  4.4.2 Price-driven Coordination for Plant-wide Control 100
4.5 Chapter Summary 105

5 Extension to Large-scale Mixed-integer Programming 106

5.1 Background 107

5.1.1 Complexity Issues in MIP 108

5.1.2 Decomposition Strategies for Mixed-integer Programming 108

5.1.3 Subgradient Optimization Techniques 110

5.1.4 Primal Solution Recovery 111

5.2 Enhancements on Subgradient Algorithms 112

5.2.1 Improved Subgradient Search Directions 113

5.2.2 An Enhanced Subgradient Algorithm 121

5.2.3 Discussions 126

5.3 Complexity Study 127

5.3.1 Theoretical Analysis 128

5.3.2 Empirical Studies 128

5.4 Approaches to Primal Solution Recovery 135

5.4.1 Primal Solution Recovery Heuristics 136

5.4.2 A Decentralized Optimization Framework 139

5.5 Case Study: Truck Allocation Problem 142

5.5.1 Application Problem 142

5.5.2 Decentralized Optimization with Coordination 144

5.5.3 Illustrative Case Study 146

5.5.4 Discussions 150

5.6 Chapter Summary 152

6 Conclusions and Future Work 154

6.1 Summary and Conclusions 154

6.2 Suggestions for Future Work 157

Bibliography 161

A Structural Empirical Studies 171

A.1 Dantzig-Wolfe Decomposition Algorithm 171

A.1.1 Linear Programming Problem Instance Generation 171

A.1.2 Monte Carlo Simulations 172

A.2 Price-driven Coordination Algorithm 173

A.2.1 Quadratic Programming Problem Instance Generation 173

A.2.2 Monte Carlo Simulations 175

A.3 Improved Subgradient Optimization Algorithm 177

A.3.1 Binary Integer Programming Problem Instance Generation 177

A.3.2 Monte Carlo Simulations 178

List of Tables

3.1 Performance of different price-update strategies 52

4.1 Performance comparison of different plant-wide MPC target calculation approaches for interacting MIMO systems 94
4.2 Performance comparison of different plant-wide MPC target calculation approaches for interacting MIMO unit network 97
4.3 Important CV weightings 101
4.4 QP-based MPC target calculation performance comparison 102

5.1 Parameters for trucking 149
5.2 Other parameters 149
5.3 Different optimization strategies for truck allocation problem 149

List of Figures

2.1 D-W decomposition mechanism 17

2.2 Worst-case behavior 26

2.3 D-W decomposition: CCN vs. number of linking constraints 31

2.4 D-W decomposition: computational performance vs. number of linking constraints 32

2.5 D-W decomposition: CCN vs. subproblem size 32

2.6 D-W decomposition: computational performance vs. problem size 33

2.7 D-W decomposition: CCN vs. number of subproblems 34

2.8 D-W decomposition: computational performance vs. number of subproblems 34

2.9 D-W decomposition: CCN vs. RSR 35

2.10 D-W decomposition: computational performance vs. RSR 36

2.11 D-W decomposition: computational dominance of Max subproblem 37

3.1 Limitations of D-W decomposition for QP 44

3.2 Infomation Flow in Price-driven Coordination 47

3.3 Convergence of different price update strategies 53

3.4 Determination of equilibrium price from supply and demand curves 54

3.5 Interior case: CCN vs. number of linking constraints 63

3.6 Boundary case: CCN vs. number of linking constraints 63

3.7 Interior case: computational performance vs. number of linking constraints 64

3.8 Boundary case: computational performance vs. number of linking constraints 64

3.9 Interior case: computational time distribution vs. number of linking constraints 65

3.10 Boundary case: computational time distribution vs. number of linking constraints 66

3.11 Interior case: CCN vs. subproblem size 66

3.12 Boundary case: CCN vs. subproblem size 67

3.13 Interior case: computational performance vs. problem size 67

3.14 Boundary case: computational performance vs. problem size 68

3.15 Interior case: computational time distribution vs. problem size 68

3.16 Boundary case: computational time distribution vs. problem size 69

3.17 Interior case: CCN vs. number of subproblems 70

3.18 Boundary case: CCN vs. number of subproblems 70

3.19 Interior case: computational performance vs. number of subproblems . . . 71

3.20 Boundary case: computational performance vs. number of subproblems . . 71

3.21 Interior case: computational time distribution vs. number of subproblems . 72

3.22 Boundary case: computational time distribution vs. number of subproblems 72

3.23 Interior case: CCN vs. RSR 73

3.24 Boundary case: CCN vs. RSR 74

3.25 Interior case: computational performance vs. RSR 74

3.26 Boundary case: computational performance vs. RSR 75

3.27 Interior case: computational time distribution vs. RSR 75

3.28 Boundary case: computational time distribution vs. RSR 76

3.29 Boundary case: CCN vs. number of active constraints 76

4.1 Bi-level MPC technology 81

4.2 Decentralized MPC 84

4.3 Coordinated, decentralized MPC 87

4.4 Demonstration of interstream consistency 88

4.5 Interacting MIMO operating units 90

4.6 Calculated targets by different approaches 94

4.7 An interacting MIMO unit network 96

4.8 Pulp mill benchmark process 99
4.9 Closed-loop response 1: solid line (coordinated); dash line (decentralized) . 103
4.10 Closed-loop response 2: solid line (coordinated); dash line (decentralized) . 103

5.1 Weighting function dynamics 114
5.2 Geometric interpretation - obtuse angle 115
5.3 Adjustment of subgradient direction 118
5.4 Improved subgradient algorithm - part I 122
5.5 Improved subgradient algorithm - part II 123
5.6 Subgradient optimization: CCN vs. number of linking constraints 130
5.7 Subgradient optimization: computational performance vs. number of linking constraints 130
5.8 Subgradient optimization: CCN vs. subproblem size 131
5.9 Subgradient optimization: computational performance vs. subproblem size 132
5.10 Subgradient optimization: CCN vs. number of subproblems 132
5.11 Subgradient optimization: computational performance vs. number of subproblems 133
5.12 Subgradient optimization: CCN vs. RSR 134
5.13 Subgradient optimization: computational performance vs. RSR 134
5.14 Basic idea of interior path methods 136
5.15 Interior path using subgradient lower bound 137
5.16 Diagram of a generalized coordination framework 140
5.17 Decentralized framework for MIP 141
5.18 Truck-and-shovel oil sands mining operations 142
5.19 Model-based coordination framework 151

6.1 Geometric interpretation - acute angle 160

"A journey of a thousand miles begins with a single step."

– by Lao Tzu

1

Introduction

Quite often, high performance operation of industrial plants or business units can be achieved by applying optimization techniques. Large-scale optimization problems frequently occur in such large-scale operations and may have special structure. Decomposition and coordination strategies are good candidates for solving such large-scale optimization problems, and sometimes are mandatory for truly large problems, which cannot otherwise be solved because of time and/or storage limitations. This thesis mainly contributes to developing more efficient decomposition and coordination methods to provide on-line solution of three typical classes of large-scale optimization problems (*i.e.*, linear programming (LP), Quadratic programming (QP) and Mixed-integer programming (MILP) problems).

1.1 Overview of Large-scale Optimization

Large-scale operations optimization problems frequently occur in the process industries. With the growing understanding of the physical and chemical phenomena underlying processes at the macro, micro, and nano scales, more complicated models can be developed for industrial applications. These detailed models can represent the plant more closely. In general, it is considered desirable to use more accurate, higher fidelity models in operations optimization to improve control and optimization performance. Such higher fidelity models are usually more complex and larger scale.

Although computer speed increases, on-line solution to large-scale optimization problems as a whole is very expensive and difficult, if not impossible. Furthermore, the scalability of solvers, which is related to problem size, and the complexity of solvers, which is related to the implementation of a method, are bottlenecks for real-time optimization in large-scale industrial operations. These have limited the scope and fidelity of many industry-scale applications. Therefore development of computationally efficient methods to solve large-scale optimization problems is of great significance in industrial applications.

1.1.1 Optimization of Large-scale Operations

In modern industry, more and more operations are integrated for various purposes and thus result in real large-scale systems. Such large-scale systems can be decomposed into a number of interconnected subsystems either for conceptual or computational reasons (Siljak, 1991). As engineering optimization has been one of the most effective means of gaining a high level of operation's performance, large-scale optimization problems considered here are of high dimension with respect to the number of variables involved in the problem formulation and have decomposable or separable structure for the purpose of computation or in nature of the problem itself (Jamshidi, 1983; Wismer, 1971).

Large-scale optimization problems appear frequently in plant-wide process design, control, and operations optimization. The simultaneous design and control of processes can be performed by optimizing large-scale, complex dynamic models involving discrete and continuous decision variables, time-varying disturbances and parametric uncertainties

(Mohideen *et al.*, 1997; Bansal *et al.*, 2000). In the literature, typical examples of large-scale process control problems can be found such as decentralized supervisory control with communicating controllers (Barrett and Lafortune, 2000); modeling and control of an N-stand cold rolling mill (Jamshidi, 1983); and plant-wide distributed model predictive control (Camponogara *et al.*, 2002). Furthermore, the integration of existing large-scale optimization problems results in a higher dimensional optimization problem. For instance, the leading industry control companies have identified the opportunities and trends to integrate plant-wide Advance Process Control (APC), Real-time Optimization (RTO) and Planning and Scheduling (Havlena and Lu, 2005).

Good surveys of large-scale operations optimization can be found in Floudas (1995) and Goux and Leyffer (2001), which introduced extensive research interest and progress in process synthesis, batch plant design, and cyclic scheduling in chemical engineering applications. When the problems are formulated as non-linear and/or mixed-integer programming problems, they become more difficult (than linear programs) and can often be said to be large-scale in terms of computational requirements. Moreover, large-scale optimization problems are also frequently encountered in the operation of transportation companies, such as the optimal scheduling and assignment of fleet and staff. The air fleet assignment problem is to determine a "best" match between the type of aircrafts and the flight segment, given a flight schedule and set of available aircrafts (Hane *et al.*, 1995). The vehicle routing and crew scheduling (VRCS) problems consider both vehicle routing and crew scheduling problems simultaneously (Desaulniers *et al.*, 2001), where the VRCS problems can give rise to very large-scale mixed-integer programming problems. More and more examples are and will be added to the list of applications of large-scale operations optimization, such as the water resources management problems in Cai *et al.* (2001), all of which show continuous intensive interest in this area in recent research.

In this thesis, the optimization is through mathematical programming, which is based on a first-principles model or an empirical model that describes process operations in sufficient detail to serve specified purposes.

1.1.2 Solving Large-scale Optimization Problems

Depending on specific requirements, such as the solution time (or computational efficiency) and accuracy (or optimality), there are three approaches to solving large-scale operations optimization problems, which can be classified as centralized, decentralized, and coordinated, decentralized optimization schemes [1]. As a large-scale system is decomposed into a number of interconnected subsystems either for conceptual or computational reasons, one difference between these optimization schemes is in how the interactions or connections between subsystems are handled. The centralized scheme explicitly considers all interactions between subsystems, while the (fully) decentralized scheme ignores all the interactions. These two schemes represent the two extremes with respect to the best achievable performance and computational requirements (Cheng *et al.*, 2004). The coordinated optimization scheme represents a trade-off between the above two extremes, wherein the subproblems are coordinated by considering important interactions.

Centralized optimization methods are usually designed for and implemented on a single computer and they do not take advantage of special problem structure (e.g., block-angular structure, stairwise structure) that can reduce computational expense. For instance, in the refinery production planning and scheduling applications (Gothe-Lundgren *et al.*, 2002; Lababidi *et al.*, 2002), a large single model for the entire plant operations (including multiple operating units) is assembled and integrated within a centralized optimization system, which is involved with a large mixed-integer programming problem.

For large-scale optimization problems, the centralized optimization schemes may require high-efficiency optimization solvers and high-performance computing environment. At the problem formulation stage, inherent special structure of the problem, such as the decomposability of the overall operations into individual unit operations, may not be considered since the centralized optimization algorithms usually do not take advantage of such special structure.

In practice, the amount of computational effort required to solve an optimization problem usually grows much faster than the size of the corresponding system or operation. Therefore, within the centralized optimization framework, the problems arising in large-scale operations (e.g., a refinery) may become either impossible or impractical to solve even with high-performance computing machines. Another potential deficiency is the reliability of such a centralized optimization system, which may provide an "all-or-nothing" behavior, i.e., such a system may provide either complete success or failure, with no appropriate strategies to mitigate the failure.

---

[1] This is often referred to as a "coordinated optimization scheme" in this thesis.

(Fully) decentralized optimization are commonly used approaches to large-scale operations optimization in industrial practice, especially when the problem can be "naturally" decomposed. An example is the unit-wide local operations optimization (e.g., a real-time optimization (RTO) system) for a refinery process unit (e.g., a vacuum distillation unit, hydrocracker unit, coking unit,and amine gas treaters, *etc.*). In this case, independent multiple RTO systems may be used within a refinery wide. These methods are usually implemented on a distributed computing environment. To formulate a decentralized optimization problem for large-scale operations, modular (decentralized) modeling methods can be used, with interactions among individual unit models excluded[2] in the optimization. A set of subproblems can be solved simultaneously, using solvers that can be selected according to the dimension of decentralized models (sub-problem sizes). Therefore, high-efficiency optimization solvers and high-performance computing environments are not a necessity.

Different from the centralized optimization opproaches, the problem structure should be well used in the decentralized optimization schemes. Because of the use of decentralized computing environment, the computational load can be well balanced through appropriate subproblem formulation. In addition, the determination of the values of interaction variables is of great importance to gain suboptimal operations close to the true optimum, and to ensure feasibility of the operations optimization.

The decentralized optimization scheme is a more reliable approach for industrial applications, i.e., when some of the subproblem optimizers fail others can still function; and it requires reasonable computational load for subproblems due to the decomposition of large-scale problems. However, this scheme usually provides sub-optimal solutions instead of the optimal solution, and sometimes ignoring the interactions may result in an infeasible operations.

---

[2] The variables representing interactions between unit operations are usually fixed at some values within the execution of optimization, and can be updated within the interval of two optimization executions.

Coordinated, decentralized optimization approaches aim to gain a good trade-off between the above two optimization approaches. Two strategies can be used for the modeling: (1) centralized modeling, decomposition and coordination; (2) decentralized modeling, incorporating linking constraints[3], and then coordination. In the coordinated, decentralized optimization framework, usually a coordination problem plus a group of subproblems are solved iteratively within an optimization execution. The coordinator, by solving the coordination problem, deals with the linking constraints and local solutions from subproblems, and thus drives the local optimal operations to the overall optimum. It should be noted that this structure requires a computing node for the coordinator in addition to the nodes required by a decentralized optimization system.

In the past few decades, we have seen the identification of many classes of structured problems and the development of many algorithms for their solution. In the period initiated by Dantzig-Wolfe decomposition for solving large-scale linear programs (Dantzig and Wolfe, 1960), Benders' partitioning algorithm was developed for solving mixed-integer programming problems (Benders, 1962), then later Rosen's partitioning method was developed to solve a primal-dual pair of problems which have block-angular structure (Lasdon, 2002; Rosen, 1964). The survey by Molina (1979) gave a good summary of the best decomposition and coordination strategies for solving large-scale optimization problems. The Dantzig-Wolfe method can be extended to solve convex nonlinear problems (Whinston, 1966). Pigot (1964) published a "double decomposition method" that could be thought as a combination of the Dantzig-Wolfe and Benders approaches to give upper and lower bounds of the optimal solution, but it has more complicated implementation. After the publication of article "Two Level Planning" (Kornai and Liptak, 1965), more interest has been shown in the decomposition of a mathematical programming problem. For example, Sanders (1965) published a method of primal decomposition of nonlinear programming problems using the Lagrange multipliers. Recently, an auction-based/price-driven coordination method was developed in Jose and Ungar (1998*a*; 1998*b*) on the basis of the work of Jennergren (1973), which can handle situations where the Lagrangian-based method may be inadequate, by using so called "slack resource" auction. The *decomposition and dynamic cut generation* (DDCG) (Ralphs and Galati, 2003) decomposes an MIP problem into an LP master and a pure IP subproblem.

---

[3] The constraints that describe the interactions involving multiple operating units.

Obviously, improvement of the coordinated optimization scheme can be made through enhancing both the coordinator and subproblem solvers. This thesis focuses on the study of efficient coordination mechanisms, i.e., an efficient way to make use of linking constraints and subproblem solutions, with an assumption that the subproblems can be solved with appropriate optimization solvers.

The coordinated optimization system can provide similar reliability as the decentralized optimization system, because a minor modification to the existing decentralized optimization scheme is accomplished by adding an extra node to the distributed computing network. Whenever the coordinator fails, the optimization system becomes the original decentralized optimization approach. In terms of the solution quality, the coordinated optimization scheme can provide an optimal, or nearly so, solution, depending on how much interaction is considered by the linking constraints. From the perspective of investment cost, this approach also implies significant benefit because the resulting optimization system can be built on a previous decentralized optimization system, by adding just one additional computing node. To satisfy the increasing requirement of computation, one way is to develop more efficient coordination mechanism. Thus a coordinated, decentralized approach can address the computational challenges that arise as problem scope and complexity increases.

1.2 Research Scope

As discussed in the previous section, this thesis concentrates on the development of a coordinated, decentralized optimization framework, which takes advantage of decomposition and coordination optimization strategies, for solving large-scale operations optimization problems. In particular, this work emphasizes the development, complexity analysis and applications of coordination mechanisms for linear programming (LP), quadratic programming (QP) and mixed-integer linear programming (MILP) problems, which represent typical classes of optimization problems encountered in industrial operations[4]. Based on a thorough understanding of the best available decomposition and coordination strategies, this work has made efforts to improve the computational efficiency of the existing optimization strategies to satisfy the specified on-line solution requirements, by enhancing the corresponding coordination mechanism. To investigate the feasibility of applying the proposed coordination optimization framework to real large-scale problems, complexity analysis and computational studies are carried out to study the scalability of the proposed optimization algorithms.

In the area of process control, as model predictive control (MPC) often utilizes LP and QP optimization techniques, this thesis discusses the application of the proposed coordinated, decentralized optimization framework for large-scale MPC, e.g., plant-wide control and optimization. Lu (2003) has discussed that the coordination/integration of decentralized controllers is a challenging large-scale optimization problem recently. Model predictive control has incorporated both control and optimization techniques, and its wide application establishes a solid foundation for plant-wide control and optimization. For large-scale process control problems, the desired characteristics of an MPC scheme includes: high-level performance – solutions comparable to the best achievable optimum (Cheng *et al.*, 2004); high degree of reliability – the capability that some control subsystems or portions thereof are able to function when other subsystems fail; good operability – the ease of the implementation of the algorithms; and excellent flexibility or extendability – the ease of the modification to functional capacity of the system. The commonly used strategies for plant-wide MPC control and optimization are centralized schemes and decentralized schemes, which represent the two extremes in the trade-off among the above desired characteristics.

In addition, this work also investigates the applicability of the proposed coordinated optimization framework in mining operations optimization, by solving the truck allocation problem in truck-and-shovel mining operations. In this case, the truck allocation problem, which can involve multiple operational parties (e.g., ore hauling, overburden removal and maintenance), is formulated as an MIP problem with decomposable block-wise structure. This work illustrates a practical approach to this problem with appropriate implementation of the proposed coordinated optimization scheme.

---

[4] Although this work is focused on mixed-integer linear programming (MILP) and binary integer programming (BIP) problems, the proposed decomposition and coordination algorithms (i.e., subgradient optimization algorithms) are applicable to general MIP problems as well, e.g., a mixed-integer nonlinear programming (MINLP) problem

For existing decentralized optimization schemes, it is assumed that each subsystem has its own computing node to perform optimization calculations. To facilitate the development of this work, synchronization assumptions for subsystem optimizers have been made[5]. For example, in plant-wide MPC applications, a common sampling time and control interval should be determined for coordination by taking into account both the fast modes and slow modes in the processes of an overall plant. Thus, it is assumed that the coordination is only active within an optimization interval. This means we are replacing a monolithic optimization calculation with a coordinated, decentralized calculation. We believe this is the first step in creating a truly decentralized set of autonomous optimizers that act in a cooperative manner.

In brief, this work addresses the problems of how to develop a coordination framework to improve the overall performance with minor modifications to the original system given a decomposable flowsheet and existing decentralized/unit optimization systems. By drawing on ideas from decentralized planning in economics, a novel methodology to design coordination systems for current decentralized optimization systems is developed. The proposed coordinated, decentralized scheme can provide a higher level of performance with a minor modification to the existing decentralized control structure. In the coordinated, decentralized optimization scheme, an additional computer will be implemented to perform coordination of individual control or optimization calculations. Therefore, similar reliability and extendability to the decentralized scheme can still be maintained. Furthermore, as the decomposition strategies use sensitivity information within the coordination mechanism, the flow of information within the optimization system can reveal how the coordination system drives the individual local optima to an overall plant-wide optimum. Thus the proposed procedure also helps understand the necessary communication between the coordinator and subsystems. In addition, this research may also provide insights into the application of decomposition strategies for solving large-scale (MI)NLP, which represents another important class of engineering optimization problems.

---

[5] We have not addressed asynchronous coordination of multiple optimization systems in this thesis.

1.3 Thesis Overview

In this work, with a general coordinated, decentralized optimization framework, coordination mechanisms are developed for solving several typical classes of optimization problems, respectively. This thesis begins with an introduction to this research, where the major objectives and research scope are stated.

In Chapter 2, the best available decomposition strategies for large-scale linear programming are investigated. In particular, the Dantzig-Wolfe decomposition algorithm (Dantzig and Wolfe, 1960) is shown to be suitable for solving the class of optimization problems of interest. A complexity study shows it has good scaling behavior and strong potential for use in the coordinated optimization framework.

Chapter 3 discusses the decomposition and coordination strategies for QP problems, which is a special case of nonlinear programming (NLP). A novel price adjustment scheme is developed by using Newton's Method, and it significantly improves the computational efficiency of the auction-based (or price-driven) coordination method. Computational studies are then carried out to investigate the scaling behavior of the price-driven coordination method.

In Chapter 4, several case studies are performed to illustrate the application of the proposed coordinated optimization framework in plant-wide MPC. Using the decomposition and coordination strategies studied in the previous two chapters, the coordinated, decentralized optimization is shown significantly improve the plant-wide operations in all the case studies.

As an extension of the work in continuous optimization, Chapter 5 discusses decomposition and coordination strategies for large-scale discrete optimization problems, in particular, mixed-integer linear programming (MILP) problems. Existing optimization methods for MILP usually contain heuristics. This work proposes a three-phase MILP decomposition algorithm within a decentralized optimization framework. The first stage incorporates the best known heuristics, whose efficiency has been proved in the literature; while for the second stage, a new heuristic is proposed for primal solution recovery. With a complexity study, the performance of the proposed MILP subgradient algorithm is tested through empirical studies. The case study of a truck allocation problem in mining operations optimization illustrates the feasibility of the application of the decentralized optimization framework.

Chapter 6 contains conclusions for this thesis and suggestions for future work.

"All truths are easy to understand once they are discovered; the point is to discover them."

– by Galileo Galilei

2

Decomposition Strategies for Large-scale Linear Programming

Decomposition approaches to solving large-scale linear programs (LP), which represent a typical class of large-scale operations problems, are discussed in this chapter.[1] Through an investigation of the best available decomposition strategies for solving large-scale linear programming problems, Dantzig-wolfe decomposition is identified as a good candidate for solving high dimensional linear programming problems with block-wise structure. By taking advantage of distributed computing environment, the Dantzig-Wolfe decomposition can provide efficient solution to decomposable linear programming problems. By using a structural complexity analysis approach, the scaling behavior of Dantzig-Wolfe decomposition algorithms is studied. The complexity study implies that, for large-scale linear programming problems with special structure, the Dantzig-Wolfe decomposition algorithm can outperform centralized optimization solvers when appropriate implementation is conducted.

---

[1] Part of this chapter has been published in Cheng *et al.* (2004) and Cheng *et al.* (2006*a*) submitted to *Computers & Chemical Engineering.*

2.1 Background

Linear programming (or, linear optimization) is a fundamental method in mathematical optimization. For a linear programming (LP) problem, the objective function and the constraints are linear functions of the decision variables. Besides its continuous development in the first half of last century, the first comprehensive theory was developed by Dantzig (1963), including geometrical analysis, duality theory, and the Simplex method. In the past few decades, an enormous number of applications of linear programming have appeared in economics, engineering, science and many other fields, from its early applications in the military resource allocation during the World War II (Rao, 1998) to its recent applications in auction pricing and proving theorems for computing science (Ye, 2006).

In practice, many large-scale LP application problems have special structure, which can be exploited by decomposition and coordination methods to provide an efficient solution. Next, the problems of our interest will be described, and the suitable decomposition algorithms will be discussed.

2.1.1 Linear Programming

In many optimization books (Chvatal, 1983; Lasdon, 2002; Dantzig and Thapa, 2002; Vanderbei, 2001), linear programming is considered classical content. It is not intended to repeat the description of a linear programming problem, but to show our interest in solving a class of LP problems with special structure. Consider the following LP problem:

$$\begin{aligned}
&\min \quad c_1x_1 + c_2x_2 + \ldots + c_nx_n \\
&\text{subject to} \\
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \le b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \le b_2 \qquad (2.1)\\
&\ldots\ldots\ldots \\
&a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \le b_m \\
&x_1, x_2, \ldots, x_n \ge 0
\end{aligned}$$

In this description, we use m to denote the number of constraints, and n to denote the number of decision variables. When represented by matrices and vectors, the LP problem can be expressed as:

$$\begin{aligned}
&\min \quad \mathbf{c}^T\mathbf{x} \\
&\text{subject to} \\
&\mathbf{Ax} \le \mathbf{b}, \quad \mathbf{x} \ge \mathbf{0} \qquad (2.2)
\end{aligned}$$

where the coefficient matrix $\mathbf{A}$ has a dimension of $m \times n$, and the cost coefficient vector $\mathbf{c}$ and decision variable vector $\mathbf{x}$ are n-vectors. In many practical applications, especially with a large-scale LP problem, such as the multi-plant production and distribution problems (Lasdon, 2002), matrix $\mathbf{A}$ takes special structure, e.g. the block-wise (or block-angular) structure:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \ldots & \mathbf{A}_{1p} \\ \mathbf{A}_1 & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2 & \ldots & \mathbf{0} \\ \ldots & \ldots & \ldots & \ldots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{A}_p \end{bmatrix} \qquad (2.3)$$

where $\mathbf{A}_{1j}$ has a dimension of $m_0 \times n_j$, and $\mathbf{A}_j$ has a dimension of $m_j \times n_j$, for $j = 1, 2, \ldots, p$. Thus, the elements of the vectors $\mathbf{c}$ and $\mathbf{x}$ can be re-organized into p

subsets, which satisfy $\sum_{j=1}^{p} n_j = n$ and $\sum_{j=1}^{p} m_j + m_0 = m$. As a result, the LP problem in (2.2) can be expressed as:

$$
\begin{aligned}
&\min \quad \mathbf{c}_1^T\mathbf{x}_1 + \mathbf{c}_2^T\mathbf{x}_2 + \ldots + \mathbf{c}_p^T\mathbf{x}_p \\
&\text{subject to} \\
&\mathbf{A}_{11}\mathbf{x}_1 + \mathbf{A}_{12}\mathbf{x}_2 + \ldots + \mathbf{A}_{1p}\mathbf{x}_p \leq \mathbf{b}_0 \qquad (2.4) \\
&\mathbf{A}_1\mathbf{x}_1 \qquad\qquad\qquad\qquad\qquad \leq \mathbf{b}_1 \\
&\qquad\quad \mathbf{A}_2\mathbf{x}_2 \qquad\qquad\qquad\quad \leq \mathbf{b}_2 \\
&\qquad\qquad \cdots\cdots\cdots \\
&\qquad\qquad\qquad\qquad\qquad \mathbf{A}_p\mathbf{x}_p \leq \mathbf{b}_p \\
&\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_p \geq \mathbf{0}
\end{aligned}
$$

In this case, excluding the constraints (2.4), the original problem can be decomposed into p LP subproblems. In this chapter, some typical techniques that can take advantage of this block-wise structure are investigated.

2.1.2 Dantzig-Wolfe Decomposition Principle

The Dantzig-Wolfe decomposition method (Chvatal, 1983; Dantzig and Thapa, 2002) is the most representative of all decomposition methods. Other typical decomposition strategies for LP problems include Benders' decomposition and Rosen's decomposition algorithms (Lasdon, 2002), as well as the *double decomposition method* by Darnel Pigot (Molina, 1979).

Although it was originally developed for solving mixed-integer programming problems, Benders' decomposition can be viewed as the dual of the Dantzig-Wolfe decomposition in terms of linear programming. Rosen's partitioning/decomposition procedure is another decomposition algorithm for linear programming and has been widely used to treat angular or dual-angular problems with both coupling constraints and coupling variables. Rosen's method uses the partition principle of fixing some complicating variables and therefore decoupling the problem into several independent lower dimensional subproblems

(Molina, 1979). The double decomposition method tackles the linear programming problems in both primal and dual spaces. In a sense, it combines both Dantzig-Wolfe decomposition and Benders' decomposition approaches, and gives upper and lower bounds of the optimal solution (Molina, 1979). As will be discussed later, many industrial application problems present suitable structures for Dantzig-Wolfe decomposition, thus discussion will be concentrated on the Dantzig-Wolfe decomposition techniques and its applications.

The Dantzig and Wolfe (1960) decomposition principle is depicted in Figure 2.1. In this decomposition approach, a large-scale linear programming problem can be separated into independent subproblems, which are coordinated by a master problem (MP). The solution to the original large-scale problem can be shown to be equivalent to solving the subproblems and the MP through a finite number of iterations (Dantzig and Thapa, 2002). Within each iteration, the MP handles the linking constraints that connect the subproblems, using information $[f_i, \mathbf{u}_i]$ supplied by the subproblems. Note that f_i is the objective function value and $\mathbf{u}_i$ is the solution of the i^{th} subproblem. Then, the MP sends its solution $[\boldsymbol{\pi}, \gamma_i]$ as price multipliers to all the subproblems for updating their objective functions. Subsequently, the subproblems with updated objective functions are re-solved. The iterative procedure continues until convergence. It can be shown that the solution at the convergence is equal to the optimal solution of the original large-scale problem (Dantzig and Thapa, 2002).

The Dantzig-Wolfe decomposition hinges on the theorem of convex combination and column generation techniques (Lasdon, 2002; Dantzig and Thapa, 2002). The theorem of convex combination, or D-W transformation, states that an arbitrary point $\mathbf{x}$ in a convex polyhedral set $\mathcal{X} = \{\mathbf{x} | \mathbf{A}\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ can be written as a convex combination of the extreme points of $\mathcal{X}$ plus a nonnegative linear combination of the extreme rays (normalized homogeneous solutions) of $\mathcal{X}$, or:

$$\mathbf{x} = \sum_{i=1}^{L} \alpha_i \mathbf{u}^i + \sum_{j=1}^{M} \beta_j \boldsymbol{\nu}^j \tag{2.5}$$

$$\sum_{i=1}^{L} \alpha_i = 1, \qquad \alpha_i, \beta_j \geq 0 \tag{2.6}$$

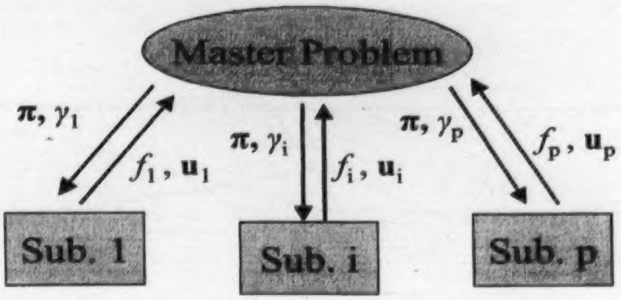


Figure 2.1: D-W decomposition mechanism

where $\mathbf{u}^i$ and $\boldsymbol{\nu}^i$ are the finite set of all extreme points and the finite set of all normalized extreme homogeneous solutions respectively. If the feasible region is bounded, we can reformulate the problem by using the extreme points only.

Although any large-scale linear program problem can be decomposed and solved by Dantzig-Wolfe decomposition (Chvatal, 1983), the approach is particularly powerful for structured linear programs. Consider a block-wise linear programming problem that has been converted to Simplex standard form:

$$\min \quad z_1 = \sum_{i=1}^{p} \mathbf{c}_i^T \mathbf{x}_i$$

subject to

$$\sum_{i=1}^{p} \mathbf{A}_i \mathbf{x}_i = \mathbf{b}_0 \tag{2.7}$$

$$\mathbf{B}_i \mathbf{x}_i = \mathbf{b}_i \tag{2.8}$$

$$\mathbf{x}_i \geq \mathbf{0} \quad i = 1, 2, ..., p$$

where (2.7) represents the linking constraints associated with p subproblems, and the constraints in (2.8) are the local constraints of independent subproblems. Via the theorem of convex combination, the master problem (MP) can be formulated as follows using the linking constraints in (2.7) and the convex combination of extreme points from (2.8), assuming that the feasible regions of subproblems are bounded[2].

$$\min \quad z_2 = \sum_{i=1}^{p} \sum_{j=1}^{N(i)} f_{ij} \lambda_{ij}$$

subject to

$$\sum_{i=1}^{p} \sum_{j=1}^{N(i)} \mathbf{p}_{ij} \lambda_{ij} = \mathbf{b}_0 \tag{2.9}$$

$$\sum_{j=1}^{N(i)} \lambda_{ij} = 1, \quad \lambda_{ij} \geq 0, \quad i = 1, 2, ..., p \tag{2.10}$$

where *N(i)* represents the number of extreme points of the feasible region in the i^{th} LP subproblem, and

$$\mathbf{x}_i = \sum_{j=1}^{N(i)} \lambda_{ij} \mathbf{u}_i^j \tag{2.11}$$

$$f_{ij} = \mathbf{c}_i^T \mathbf{u}_i^j \tag{2.12}$$

$$\mathbf{p}_{ij} = \mathbf{A}_i \mathbf{u}_i^j \tag{2.13}$$

with $\mathbf{u}_i^j$ being the j^{th} extreme point of i^{th} subproblem.

The resulting master problem has fewer rows (i.e., $(m_0 + p)$ equality constraints) in the coefficient matrix than the original problem; however, the number of columns (i.e., associated with $\sum_{i=1}^{p} N(i)$ decision variables) in the MP is larger due to an increase in the number of variables associated with the extreme points of all subproblems. For a large-scale problem, it can be a formidable task to obtain all the extreme points and formulate a full master problem. Therefore, column generation techniques have been developed to dynamically handle the more columns in the MP.

2.2 Column Generation Techniques

If the MP is solved via the Simplex method, only the basic set is needed and it has the same number of basic variables as the number of rows. Thus, we do not need to explicitly know all the extreme points of subproblems. This leads to solving an equivalent problem, the *restricted master problem* (RMP), which can be dynamically constructed at a fixed size by incorporating column generation techniques (Gilmore and Gomory, 1961; Dantzig and Thapa, 2002). For example, in the MP described in (2.9) and (2.10), the linear programming problem obtained by dropping all but a subset $\bar{N}$ of the $N(i)$ columns associated with λ_{ij} is called an RMP of the original master problem.

---

[2] Unbounded cases are discussed in Lasdon (2002), Dantzig and Thapa (2002).

Assume that we have a starting basic feasible solution to the RMP and it has a unique optimum. The optimal solution provides us with the Simplex multipliers $[\boldsymbol{\pi}, \gamma_1, \gamma_2, ..., \gamma_p]$ for the basis in the current RMP, with $\boldsymbol{\pi}$ associated with (2.9) and γ_i with the i^{th} constraint in (2.10), respectively. Then, the subproblems are modified and solved to find the priced-out column associated with λ_{ij}:

$$f_{ij} = (\mathbf{c}_i^T - \boldsymbol{\pi}\mathbf{A}_i)\mathbf{u}_i^j - \gamma_i \tag{2.14}$$

and the i^{th} subproblem is:

$$\begin{aligned} &\min \quad z_i^0 = (\mathbf{c}_i^T - \boldsymbol{\pi}\mathbf{A}_i)\mathbf{x}_i \\ &\text{subject to} \\ &\mathbf{B}_i\mathbf{x}_i = \mathbf{b}_i \\ &\mathbf{x}_i \geq \mathbf{0} \end{aligned} \tag{2.15}$$

Here we notice that only minor modification is made to each subproblem, i.e., a term containing sensitivity information is introduced into each subproblem objective function. An optimal solution is reached when the following condition is satisfied:

$$\min_{i,j} f_{ij} = \min_{i} (z_i^0 - \gamma_i) \geq 0, \;\; i = 1, ..., p \tag{2.16}$$

A complete proof of the optimality and finite convergence can be found in Dantzig and Thapa (2002). When condition (2.16) is not satisfied, a column generation strategy is used to determine the column or columns that will enter the basis of the RMP.

The coordination of subproblems can be regarded as a procedure of directional evaluation of the feasible extreme points of the subproblems, in which the coordinator (RMP) evaluates and selects subproblem solutions under the guidance of some "rules".

The selected or priced-out extreme points will be used to generate columns needed for updating the RMP. The column generation techniques are among the "rules" that show good performance in directing the evaluation of subproblem feasible solutions. With column generation techniques, instead of an exhaustive traversal of all extreme points of the subproblems, usually only a small subset of the extreme points are required to be evaluated during the coordination procedure.

2.2.1 Single-column vs. Multi-column Generation

Several column generation techniques can be found in the literature. In the single-column generation scheme (Lasdon, 2002), the minimum objective function value of problem (2.16) is assumed to come from subproblem s ($1 \leq s \leq p$), i.e., the solution $\mathbf{x}_s(\boldsymbol{\pi})$ solves subproblem s. Then, the column to enter the basis is generated by:

$$\begin{bmatrix} \mathbf{A}_s\mathbf{x}_s(\boldsymbol{\pi}) \\ \mathbf{i}_s \end{bmatrix} \tag{2.17}$$

where $\mathbf{i}_s$ is an n-component vector with a "1" in position s and zeros elsewhere. The generated column is associated with the most favorable subproblem (i.e., that with the most negative reduced cost). Thus, with single-column generation algorithm, the updated RMP can be expressed as:

$$\min \quad z_3 = \sum_{i=1}^{p} \sum_{J_{basis}} f_{ij}\lambda_{ij} + f^*\lambda^*$$

subject to

$$\sum_{i=1}^{p} \sum_{J_{basis}} \mathbf{p}_{ij}\lambda_{ij} + \mathbf{p}^*\lambda^* = \mathbf{b}_0 \tag{2.18}$$

$$\sum_{J_{basis}} \lambda_{ij} + \lambda^* = 1, \quad i = 1, 2, ..., p \tag{2.19}$$

$$\lambda_{ij} \geq 0, \quad \lambda^* \geq 0 \tag{2.20}$$

where the λ_{ij} is the current basic variables and λ^* is the variable entering the basis. The terms associated with λ^* can be derived from (2.11) to (2.13) and (2.17), which contribute to the generated column. J_{basis} is an index set whose elements correspond to the indices of subproblem feasible solutions, which are now in the RMP basis. Particularly, in the above formulation, the number of variables in the basis is $|J_{basis}| = m_0 + p$,[3] which shows the dimension of the basic set in the RMP is $(m_0 + p)$, where m_0 is the number of linking constraints in (2.7).

It should be noted that any other subproblem with a negative reduced cost has the potential to generate a column to enter the basis of the master problem (Lasdon, 2002). To take advantage of other favorable subproblems, multiple columns could be considered for generating a new RMP. Several variants of multi-column generation techniques are discussed in Lasdon (2002) and Dantzig and Thapa (2002). In this work, we study the multi-column generation scheme suggested in Lasdon (2002). Thus, to incorporate all potential favorable proposals, a "new" column is generated in the RMP for each subsystem by applying (2.17):

$$\min \quad z_4 = \sum_{i=1}^{p} \sum_{J_{basis}} f_{ij} \lambda_{ij} + \sum_{i=1}^{p} f_i^* \lambda_i^*$$

subject to

$$\sum_{i=1}^{p} \sum_{J_{basis}} \mathbf{p}_{ij} \lambda_{ij} + \sum_{i=1}^{p} \mathbf{p}_i^* \lambda_i^* = \mathbf{b}_0 \tag{2.21}$$

$$\sum_{J_{basis}} \lambda_{ij} + \lambda_i^* = 1, \quad i = 1, 2, ..., p \tag{2.22}$$

$$\lambda_{ij} \geq 0, \quad \lambda_i^* \geq 0 \tag{2.23}$$

The above problem has p more variables than constraints, rather than one more as in the single-column generation case. Using the size of the coefficient matrix in Simplex standard form to represent the size of the problem, the RMP with multi-column generation has a size of $(m_0 + p) \times (m_0 + p + p)$ while the RMP with single-column generation has a size $(m_0 + p) \times (m_0 + p + 1)$. One would expect a greater decrease in z_4 through every iteration, and thus, a reduction in the number of iterations. As was discussed in Lasdon (2002) and verified by the computational experiments (Cheng *et al.*, 2005*a*), the advantage of having more columns in the RMP outweighs the disadvantage of increased RMP size. Since it can show higher computational efficiency, the Dantzig-Wolfe decomposition algorithm with multi-column generation is adopted in this work.

---

[3] Here the notation "| |" is used to denote the number of elements or the length of a data set as is common practice in computing science.

2.2.2 Dantzig-Wolfe Decomposition Algorithms

Depending the column generation techniques adopted, variants of Dantzig-Wolfe decomposition algorithms have been developed. With the multi-column generation technique discussed in the previous section, a bi-level decomposition algorithm can be formulated for the solution of the problem described in (2.7) to (2.8). Assuming the RMP in (2.9) to (2.10) has an initial feasible basic solution, the Simplex multipliers $[\boldsymbol{\pi}, \gamma_1, \gamma_2, ..., \gamma_p]$, with $\boldsymbol{\pi}$ associated with constraint set (2.9) and γ_i associated with (2.10), can be obtained. The following statement gives a brief description for the Dantzig-Wolfe decomposition algorithm:

- Step 1: Using the Simplex multipliers $\boldsymbol{\pi}$, solve the subproblems in (2.15) and obtain the subproblem solutions $\mathbf{x}_i(\boldsymbol{\pi})$ and optimal objective function values z_i^0. Denote $\mathbf{x}(\boldsymbol{\pi}) = [\mathbf{x}_i(\boldsymbol{\pi}), ..., \mathbf{x}_p(\boldsymbol{\pi})]$.

- Step 2: Compute $\min_{i,j} f_{ij} = \min_i (z_i^0 - \gamma_i)$. If $\min_{i,j} f_{ij} \geq 0$, an optimal solution is achieved and can be calculated from:

$$\mathbf{x}^* = [\sum_{Jbasic} \lambda_{1j}\mathbf{x}_1^j, ..., \sum_{Jbasic} \lambda_{pj}\mathbf{x}_p^j] \tag{2.24}$$

  where the points $\mathbf{x}_i^j$ are the extreme points of the i^{th} subproblem in the index set of J_{basis}; otherwise, go to Step 3.

- Step 3: If $\min_{i,j} f_{ij} < 0$, the p entering columns [4] can be calculated as:

$$\begin{bmatrix} \mathbf{A}_1\mathbf{x}_1(\boldsymbol{\pi}) \ldots \mathbf{A}_p\mathbf{x}_p(\boldsymbol{\pi}) \\ \mathbf{I} \end{bmatrix} \tag{2.25}$$

---

[4] Here, we have multiple columns, i.e., p new columns, to enter the RMP as we adopt the multi-column generation techniques; however, if single-column generation technique is used, only one column is entering the RMP.

and then added to the previous RMP as new columns. Solve the resulting RMP and obtain a new basis (by removing the leaving columns as in Simplex method) and go back to Step 1 and repeat.

2.2.3 Discussions

The higher computational efficiency produced by multi-column generation techniques has been widely discussed in the literature (e.g., (Dantzig and Thapa, 2002; Lasdon, 2002)) and verified through empirical studies (e.g., (Cheng *et al.*, 2005*a*; Cheng *et al.*, 2006*a*)). The mechanism that drives the higher efficiency of multi-column generation in comparison to the single-column generation can be explained by analogy.

Similar to the Simplex method, in Dantzig-Wolfe decomposition, different "pricing" strategies can be used to determine the entering columns (analogously, entering variables in the Simplex method). If we know all extreme points, the (full) master problem is equivalent to the original LP problem, and we only need to solve the (full) MP once to get the optimum solution. When the master problem is solved, by fully pricing all the columns, the "best" entering variable[5] can be determined. At the other extreme, in the single-column generation, the entering column corresponds to the subproblem with most negative objective function value, and as a result, only one entering column together with the existing columns in current RMP is to be priced when the resulting RMP is solved by the Simplex method. Obviously, the pricing process in the solution of the RMP with single-column generation is a partial pricing (Nash and Sofer, 1996) process. With multi-column generation, more entering variable candidates, including the one that can be generated by single-column generation, are to be priced in the resulting RMP. In the partial pricing scheme, a larger subset of the full columns is available for pricing, which may result in a better entering variable, or, at least an entering variable as good as the one generated by single-column generation.

In addition, it is noted that the RMP has a fixed size, i.e., whenever there are new columns generated as the entering columns, there will be the same number of columns dropping from the current basis of the RMP. Based on different ways of dealing with the "leaving" columns, there are variants of the Dantzig-Wolfe decomposition algorithms. In Dantzig and Thapa (2002), two variants that are discussed retain each "dropped" (non-optimal) column as a supplementary column in the current RMP until the problem size reaches some specific limit of the computer memory. Our strategy to treat the "dropped" columns is different from those two variants in that the fixed number of columns (i.e., $m_0 + 1$ in single-column generation and $m_0 + p$ in multi-column generation) for the RMP is not related to the memory limit. At this point, a subset of the columns that are priced out as the most unfavorable (with most positive objective function values) is dropped from the current RMP. If the computer memory allows, by incorporating either of the two variants discussed above, computational efficiency enhancement can be observed for the multi-column generation technique discussed in the previous section.

---

[5] For steepest descent pricing, the best entering variable is the variable corresponding to the most negative reduced cost.

2.3 Complexity Study

The computational efficiency of a coordination strategy is a key factor in determining the viability of using coordinated decentralized optimization approaches in industrial applications. In the following sections, we evaluate the computational efficiency of the Dantzig-Wolfe decomposition algorithm through an empirical investigation of complexity.

Without loss of generality, we consider a large-scale block-angular LP problem with p subproblems in its standard form

$$\max \quad \sum_i \mathbf{c}_i^T \mathbf{x}_i$$

subject to

$$\sum_i \mathbf{A}_i \mathbf{x}_i = \mathbf{b}_0 \tag{2.26}$$

$$\mathbf{B}_i \mathbf{x}_i = \mathbf{b}_i \tag{2.27}$$

$$\mathbf{x}_i \geq \mathbf{0} \quad i = 1, 2, \ldots, p \tag{2.28}$$

where vectors $\mathbf{x}_i$ $(n_i \times 1)$, $\mathbf{b}_i$ $(m_i \times 1)$, $\mathbf{b}_0$ $(m_0 \times 1)$, $\mathbf{c}_i$ $(n_i \times 1)$, and matrices $\mathbf{A}_i$ $(m_0 \times n_i)$, $\mathbf{B}_i$ $(m_i \times n_i)$ are specific to subproblem "i".

Computational complexity is the study of determing the cost of solving a numerical computing problem using a scientific algorithm. In this work, the cost of the proposed Dantzig-Wolfe decomposition approach can be interpreted as the required arithmetic or other computational operations[6].

2.3.1 Theoretical Analysis

Although the computational performance of an algorithm can be measured in several ways, the "worst-case" behavior and "average-case" behavior are two typical measures (Nash and Sofer, 1996).

Worst-case Behavior

Unlike a complete enumeration approach, the process of coordination can be viewed as a directional evaluation of subproblem extreme points in the RMP. When the multi-column generation technique is incorporated into a coordinated optimization scheme, shown in Figure 2.2, the RMP (coordinator) will evaluate a new set of p extreme points submitted by each subproblem at every iteration. Note that the new set of p extreme points in black (in the left ellipse of RMP rectangle frame) is not generated by an arbitrary combination but under the direction of coordinator (RMP). In Figure 2.2, the $(m_0 + p)$ extreme point set in gray (in the right ellipse of RMP rectangle frame) is associated with the RMP optimal basis at the previous coordination iteration. The p point set is a new set of extreme points submitted by subproblems at current iteration, and N_i is the number of extreme points of the i^{th} subproblem feasible region.

The worst-case behavior analysis depends on the LP techniques that are used for solving the RMP and subproblems. The worst-case behavior of Simplex methods is non-polynomial; whereas, the interior point methods are polynomial time algorithms. If Simplex methods are used and we take its worst-case performance, the Dantzig-Wolfe decomposition algorithm cannot be a polynomial time algorithm. Even though each subproblem can generate an (optimal) extreme point in polynomial time, using interior

---

[6] In this work, the definition of an arithmetic operation, such as an addition or a multiplication, is applied to two real numbers.

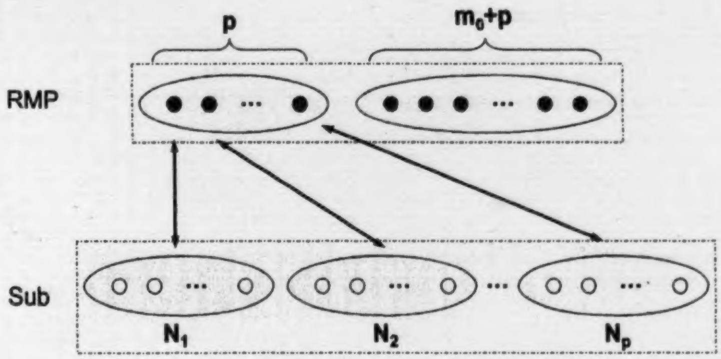


Figure 2.2: Worst-case behavior

point methods (IPM), it may take exponential time to get all the extreme points for a subproblem. Note that each subproblem has m_i constraints and n_i decision variables. Thus, the number of extreme points can be as many as the combination ${}_{n_i}C_{m_i}$. Therefore, in the worst case, whatever methods are used to solve either RMP or subproblems, the coordination process might theoretically evaluate every combination of the subproblem extreme points. Then, worst-case behavior analysis for Dantzig-Wolfe decomposition implies that it may not be a polynomial time algorithm.

Usually, only when the worst-case behavior accurately reflects the average-case behavior of an algorithm, is it used to provide an upper bound of the cost of solving a problem (Nash and Sofer, 1996). For example, the worst-case performance of interior point methods (IPM) gives a rather tight upper bound; however, a worst-case analysis does not reflect the observed performance of the Simplex method (Chvatal, 1983; Nash and Sofer, 1996). Since average-case behavior is more relevant for our work, average-case performance will be emphasized here.

Average-case Behavior

Considering the coordination mechanism discussed in the previous sections, the overall complexity for the decomposition strategy can be expressed as:

$$\mathbf{T} = \{\mathbf{T}(RMP) + \sum_{i=1}^{p} \mathbf{T}(SP_i)\} \times \mathbf{CCN} \tag{2.29}$$

where T respresents the number of arithmetic operations for solving the LP problem, and the communication cycle number (CCN) is used to distinguish the number of Simplex iterations from that of coordination iterations (the number of times to solve the RMP). Thus, the required arithmetic operations are attributed to two parts: the operations for solving the subproblems and the RMP in a single communication cycle, as well as the number of communication cycles.

If we define the first part as the non-coordination complexity:

$$\mathbf{T}(NonCo) = \mathbf{T}(RMP) + \sum_{i=1}^{p} \mathbf{T}(SP_i) \tag{2.30}$$

the overall complexity can be expressed as

$$\mathbf{T} = \mathbf{T}(NonCo) \times CCN \tag{2.31}$$

In the Simplex method, for an LP problem with $\bar{m}$ constraints and $\bar{n}$ variables in standard form, the cost of solving one iteration is $O(\bar{m}\bar{n})$ for Gaussian elimination plus $O(\bar{m}^3)$ arithmetic operations for periodic re-factorization of the basis matrix[7], thus the arithmetic operation needed in one Simplex iteration is $O(\bar{m}^3+\bar{m}\bar{n})$ (Nash and Sofer, 1996). Here, we consider the average-case performance of Simplex method[8] and take the average number of Simplex iterations as $O(\bar{m}+\bar{n})$ as in Andrei (2004). Therefore, the average behavior bound to be used is $O(\bar{m}^4+\bar{n}\bar{m}^2+\bar{m}^3\bar{n}+\bar{n}^2\bar{m})$ for the Simplex method, which shows polynomial time complexity. For the LP problem described in (2.26) to (2.28), we can derive the non-coordination complexity from:

---

[7] The $O()$ notation for a given function $g(n)$ is given as $O(g(n)) = \{f(n) : \exists\, a^+ \text{ and } n_0^+ \text{ such that } 0 \leq f(n) \leq ag(n) \text{ for all } n \geq n_0^+\}$.

[8] The observed scaling behavior of Simplex method is between $\bar{m}$ and $3\bar{m}$ (Nash and Sofer, 1996).

$$\mathbf{T}(RMP) \in O(M_0^4 + N_0 M_0^2 + M_0^3 N_0 + N_0^2 M_0) \tag{2.32}$$
$$where \quad M_0 = m_0 + p, \quad N_0 = m_0 + 2p$$

and

$$\mathbf{T}(SP_i) \in O(m_i^4 + n_i m_i^2 + m_i^3 n_i + n_i^2 m_i) \quad i = 1, 2, ..., p \tag{2.33}$$

Since the above numbers of arithmetic operations are all polynomial in corresponding m and n, the complexity of the non-coordination computation in (2.30) can be expressed as a polynomial with respect to the number of constraints and decision variables. In other words, the non-coordination arithmetic operations $\mathbf{T}(NonCo)$ can be computed very efficiently if we consider the average-case behavior of Simplex method.

The other part of the complexity analysis deals with the communication cycles. To our knowledge, there is no similar analysis in the literature and therefore resort to a study of the average behavior of CCN via a comprehensive empirical study. Next, empirical studies will be performed to investigate the computational complexity and scaling behavior of the Dantzig-Wolfe decomposition algorithm.

2.3.2 Empirical Studies

In last section, T represents the number of arithmetic operations, so equation (2.29) is used to express the theoretical (average) complexity of the algorithm (i.e., it is assumed that for each iteration the computational effort needed is in the same order); however, in the computational studies, t in equation (2.34) acts as an indicator and measurement of the complexity. So, instead of using the multiplication of CCN, the "actual" computing time at each iteration is summed up. For analysis of the computational complexity of the decomposition algorithm, since the algorithms are implemented on a sequential machine, we may express the overall complexity in terms of computational time:

$$\mathbf{t} = \sum^{CCN} \{\mathbf{t}(RMP) + \sum_{i=1}^{p} \mathbf{t}(SP_i)\} \tag{2.34}$$

where t represents the total computational time to solve a problem. Through the comparison between the computational time of each subproblem and the total computational time, we can identify and focus on the "bottleneck" subproblem, which may be the largest in dimension or hardest to solve.

Similarly, if we denote the non-coordination computational time as:

$$\mathbf{t}(NonCo) = \mathbf{t}(RMP) + \sum_{i=1}^{p} \mathbf{t}(SP_i) \tag{2.35}$$

and assume a distributed/parallel computing environment, for example, one CPU for each subproblem, then we have the equivalent computational time (parallel computing):

$$\mathbf{t}_{eqv}(NonCo) = \mathbf{t}(RMP) + \max_{i=1}^{p}\{\mathbf{t}(SP_i)\} \tag{2.36}$$

Such a distributed computing environment coincides with what is encountered in current plant-wide decentralized MPC applications. In this case, it may be desirable to balance the computational load on each computing node because the non-coordination computational complexity ($\mathbf{t}_{eqv}$) relies heavily on the largest subproblem. We will reemphasize this point in next section. Then, the computational time within a decentralized computing environment can be expressed as:

$$\mathbf{t}_{eqv} = \sum^{CCN} \mathbf{t}_{eqv}(NonCo) \tag{2.37}$$

Next the focus is on the analysis of CCN, where the relationship between CCN and the characteristic parameters of the LP problems such as m_0, p, and $|I_i| = (m_i \times n_i)$ is to be determined. Note that, each parameter may have physical meaning in real systems. For instance, in plant-wide MPC coordination, m_0 may reflect the density of interactions among operating units; p could be the number of decentralized MPC controllers or distributed industrial computers; while $|I_i|$ can represent the size of the control problems handled by an MPC subsystem. Moreover, the influence of the relative subproblem ratio (RSR), which is defined as RSR $= \{\max\frac{|I_i|}{|I_j|},\ i,j = 1,2,...,p\}$, will also be studied. The RSR gives some idea on the computational load balance throughout the distributed computing network.

In the following Monte Carlo simulations, besides the CCN, the computational efficiency and scaling behavior of Dantzig-Wolfe decomposition will also be investigated by comparing the performance between the decomposition algorithm and centralized LP solvers on the platform of MATLAB®. In both solvers, ILOG® CPLEX 9.0, which invokes the dual Simplex algorithm by default, is used to solve all LP problems. In our study, we focus on the average behavior of different optimization strategies in solving the problems with the following assumptions:

1. No cycling: many techniques can be applied to efficiently deal with cycling.
2. No degeneracy: when degeneracy occurs in practice, it can be well handled using techniques such as perturbation techniques in Simplex method.
3. The studied problems have bounded feasible regions and optimal solutions.

Due to the random nature of Monte Carlo simulation, we would like to acknowledge that there is possibility that an "average-case" problem may not be represented in our test set.

The test problem instance generation scheme is discussed in Appendix A.1.1. It randomly generates a set of LP problem instances with block-angular structure. We start from a reference problem model, whose problem size and structure should be a good reference for the comparison experiments, i.e., we can observe algorithm performance changes when we change the problem with respect to the reference model. With some preliminary tests, we choose the following set of parameters as the reference problem model:

$$\{p = 17, m_0 = 30, m_i = 40, n_i = 30 \quad i = 1, 2, ..., p\} \tag{2.38}$$

Note that the reference problem has subproblems of identical size, i.e., the problem is thus a "well-balanced" decomposable problem with $RSR = 1$. We also assume that each subproblem has been allocated to a separate distributed CPU. Therefore, the equivalent computational time t_{eqv} for the decomposition algorithm is estimated by summing up the time for solving the master problem and the most difficult subproblem, assuming a distributed computational environment. In the Monte Carlo simulation for each scenario, the number of problem instances is $200 \times 5 = 1000$, i.e., for each scenario, five runs of simulation are performed and each run solves 200 problem instances generated randomly.

Scenario 1: We fix p and $|I_i|$, change m_0 (see Appendix A.1.2). In this case, we can study the performance of decomposition and coordination with respect to the dimension of linking constraints in equation (2.26).

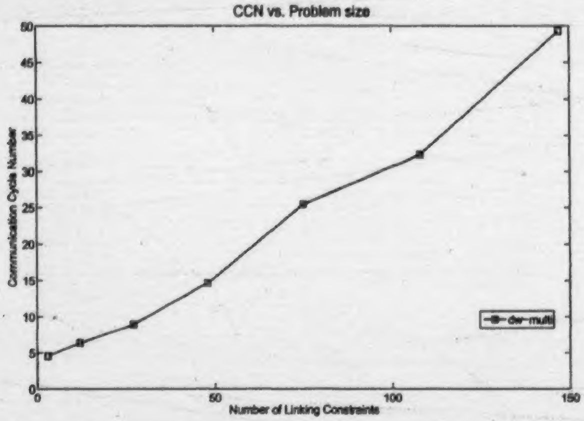


Figure 2.3: D-W decomposition: CCN vs. number of linking constraints

Figure 2.3 shows that the CCN increases almost linearly with the dimension of linking constraints increases. In Figure 2.4, when the number of linking constraints is small, the decomposition algorithm gives comparable performance to the centralized LP solver. When the number of linking constraints increases, the computational performance gets worse. This shows the computational complexity of the decomposition algorithm has strong dependence on the dimension of linking constraints.

Scenario 2: For fixed p and m_0, we change subproblem size $|I_i|$ by simultaneously changing m_i and n_i (see Appendix A.1.2). In this case, we study the algorithm performance with respect to subproblem sizes.

Figure 2.5 shows a rather surprising result. Intuitively, one may think that the CCN would increase when subproblem size increases, because the overall problem gets bigger. By analogy to the coordination of plants in a large company, larger plant decision proposals are usually less sensitive to coordination and thus do not tend to change dramatically. In such a case, the central planning board may end up with less coordination iterations. Since the solution of a larger subproblem is more time consuming, Figure 2.6 shows an increase in

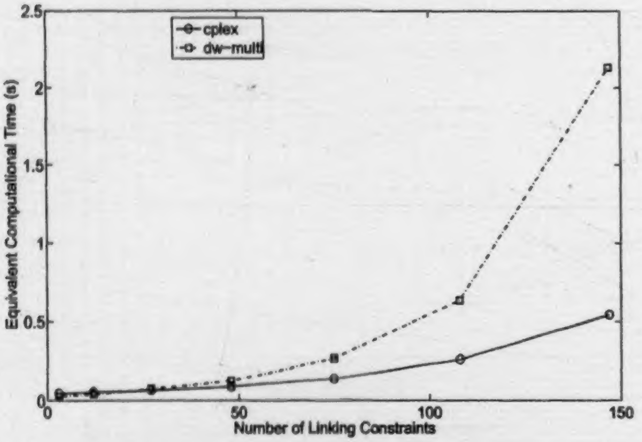


Figure 2.4: D-W decomposition: computational performance vs. number of linking constraints

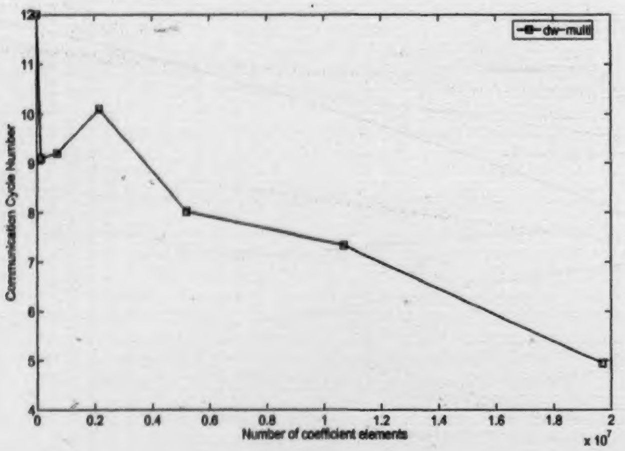


Figure 2.5: D-W decomposition: CCN vs. subproblem size

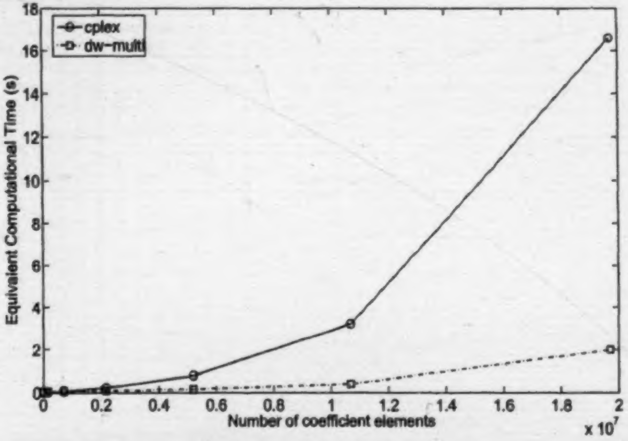


Figure 2.6: D-W decomposition: computational performance vs. problem size

the computational time of Dantzig-Wolfe decomposition algorithm, but its performance is much better than the centralized LP solver. One extreme case is that the subproblems have infinite size, then the linking constraints are negligible and the LP problem is essentially a (fully) decentralized problem, which requires no coordination (i.e., $CCN \rightarrow 0$). In addition, as the linking constraints contribute much less than the subproblems to the overall problem size, the changes of overall problem size reflect the changes of subproblem size. Thus, the overall problem coefficient number is used in the figures.

Scenario 3: We keep m_0, m_i and n_i constant, and change the number of subproblems p (see Appendix B). In this case, we investigate the performance of the coordination algorithm when more and more subproblems are integrated into the coordination system, assuming a rather well-balanced subproblem computational load.

Figure 2.7 shows that the number of subproblems p slightly influences the coordination complexity. When the number of subproblems increases, there is a minor increase in CCN. Similarly, in Figure 2.8, the number of subproblems slightly influence the computational performance when distributed computing environment is considered. In other words, the incorporation of a similar-size subsystem does not degrade the computational performance too much, which also indicates the good scaling behavior of the Dantzig-Wolfe decomposition algorithm for our class of problems.

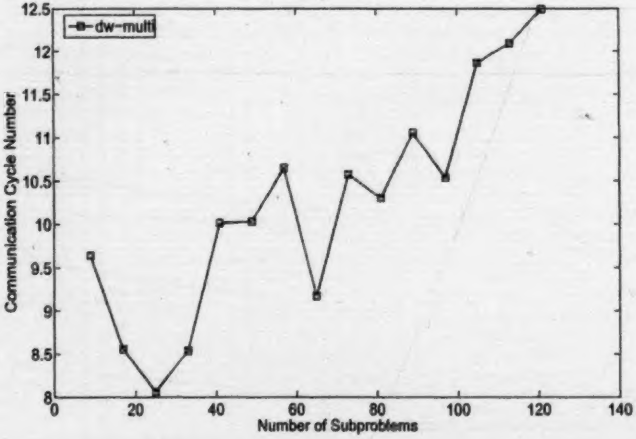


Figure 2.7: D-W decomposition: CCN vs. number of subproblems

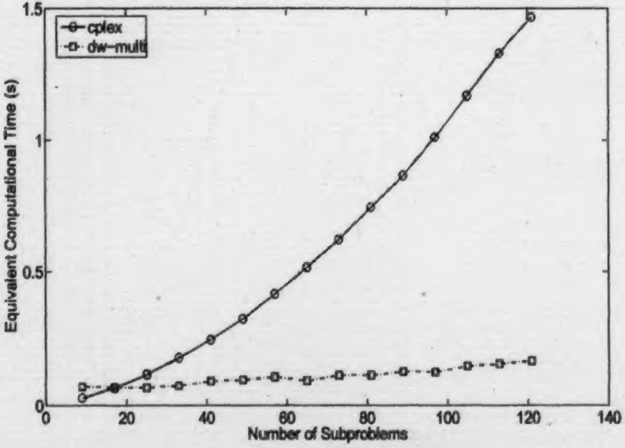


Figure 2.8: D-W decomposition: computational performance vs. number of subproblems

Scenario 4: We fix m_0, $\sum_{i=1}^{p} m_i$ and $\sum_{i=1}^{p} n_i$, i.e., we fix the overall problem size, we can study the influence of the relative subproblem ratio (RSR). In this case, we change p by combining subproblems into groups (see Appendix B) according to different partition patterns of the original LP problem. For example, when $RSR = 4$, the original 17 subproblems in the reference problem can be combined into a set of subproblems which have a problem size ratio $\{4 : 4 : 4 : 4 : 1\}$ (pattern 1) or $\{4 : 1 : ... : 1\}$ (pattern 2). The above patterns reflect two typical situations in which we have a smaller subsystem or a larger subsystem compared with others.

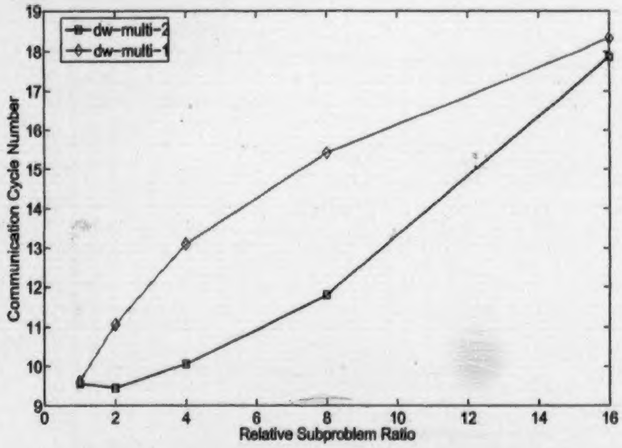


Figure 2.9: D-W decomposition: CCN vs. RSR

In Figure 2.9, the CCN of both cases is monotonically increasing as the RSR increases; while in Figure 2.10, the computational time also increases as the RSR increases. It should be noted that there are more subproblems in pattern 2 than pattern 1. When we have more subproblems, more vertices from subproblems can be incorporated into the RMP, thus a faster convergence is expected, i.e., a smaller CCN in pattern 2. In addition, as an identical CPU is assumed to be allocated to each subproblem, the number of CPUs, p_0, corresponds to the number of resulting subproblems. We can see that even for the same LP problem, different decomposition patterns lead to different computational performance. Moreover,

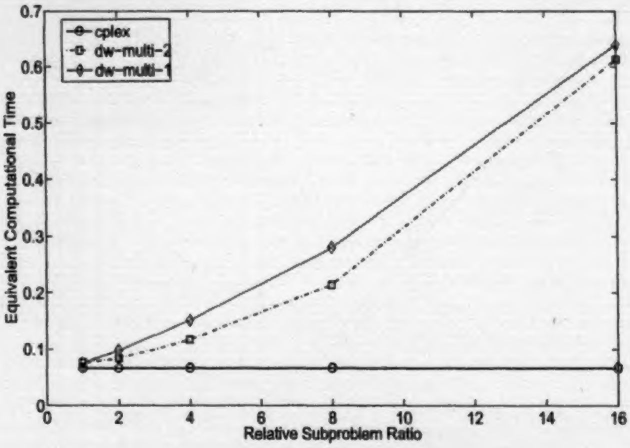


Figure 2.10: D-W decomposition: computational performance vs. RSR

if we define

$$\mathbf{D}_{subm} = \frac{\max_{i=1}^{p_0} \mathbf{t}(Sub_i)}{\sum_{i=1}^{p_0} \mathbf{t}(Sub_i)} \tag{2.39}$$

as a quantity that represents the computational time dominance of the largest subproblem, and

$$\sum_{i=1}^{p_0} \mathbf{t}(Sub_i) = \mathbf{t}(NonCo) - \mathbf{t}(RMP) \tag{2.40}$$

can be obtained from our simulation.

Shown in Figure 2.11, for the two decomposition patterns, the computational time taken by the largest subproblem dominates the computational time for solving subproblems when RSR increases. This also addresses the reason we should balance the computational load on each computing node in our experiment design.

Remarks: Although the complexity study has not yielded an accurate mathematical expression for the computational complexity, it does provide new insight and reveal some inherent features of the complexity of Dantzig-Wolfe decomposition algorithm. Moreover, this study also provides some guidelines for coordination system design with Dantzig-Wolfe decomposition principle. Advantages over a centralized LP solver are gained by using the decomposition algorithm to solve an LP problem, which has the following properties:

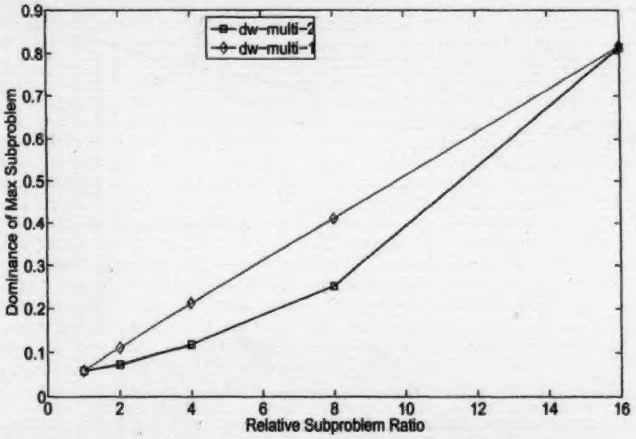


Figure 2.11: D-W decomposition: computational dominance of Max subproblem

- a block-angular structure;
- a large scale;
- relatively low-dimensional linking constraints;
- relatively high-dimensional subproblems;
- well-balanced computational load;
- large number of subproblems, if distributed computing power is available.

In many industrial applications, the plant-wide MPC target calculation problem can be formulated as an LP problem having most of the above properties, and as a result, can be efficiently solved by Dantzig-Wolfe decomposition.

2.4 Chapter Summary

In this chapter, the best available decomposition strategies for solving large-scale linear programming problems have been investigated. In particular, by taking advantage of distributed computing environment, the Dantzig-Wolfe decomposition is very suitable for solving high dimensional linear programming problems with block-wise structure.

With a well-designed structural complexity analysis approach, new insight into the relationships between computational performance and problem structural parameters was gained through theoretical analysis and a comprehensive empirical study of the scaling behavior of Dantzig-Wolfe decomposition algorithms. The complexity study shows that, for large-scale linear programming problems with special structure, the Dantzig-Wolfe decomposition algorithm can outperform centralized optimization solvers when appropriate implementation is conducted. Moreover, the complexity study also provides guidelines for the practical application of the decomposition and coordination methods.

"Truth is ever to be found in the simplicity, and not in the multiplicity and confusion of things."

– by Sir Isaac Newton

3

Decomposition Strategies for Large-Scale Quadratic Programming

---

This chapter discusses decomposition and coordination strategies for solving large-scale quadratic programming (QP) problems, which represent a typical class of optimization problems in industrial applications, such as process design, operations, and control. The focus of this chapter is on the auction-based (or price-driven) coordination methods (Jose and Ungar, 1998*a*), which was originally developed for solving general nonlinear programming problems. With appropriate partitioning, a large-scale QP problem can be equivalently converted into a set of independent subproblems linked by a coordination problem. To improve the computational efficiency of the price-driven coordination method, an efficient price adjustment scheme is proposed by using Newton's method to take advantage of the sensitivity information from subproblem solution. The proposed price adjustment scheme can significantly improve the computational efficiency of the price-driven coordination methods when solving large-scale QP problems. The structural complexity analysis is used to gain more insight into the computational performance and scaling behavior of the proposed price-driven coordination method. [1]

3.1 Background

Large-scale quadratic programming problems are frequently encountered in industrial applications. In model predictive control, the steady-state target calculation and dynamic optimization problems can be formulated as quadratic programming (QP) problems (Ying and Joseph, 1999; Qin and Badgewell, 2003). In addition, most nonlinear programming problems in process optimization are solved by sequential quadratic programming (SQP), in which a quadratic programming subproblem is solved iteratively (Edgar and Himmelblau, 2001). Further, many parameter estimation, resource assignment problems, etc., take the form of QP problems. Therefore, decomposition strategies for efficiently solving large-scale QP problems may have potential for use in a wide range of operations optimization problems.

There are several existing decomposition methods and their variants can be used for solving large-scale quadratic programming problems. These decomposition methods share a basic mechanism: the original problem is decomposed into smaller subproblems, which are coordinated by a so called "master problem".

3.1.1 Decomposition Strategies for Nonlinear Programming

The Dantzig-Wolfe method can be extended to solve convex nonlinear problems (Molina, 1979; Whinston, 1966), among which QP problems are the simplest. This approach has very elegant economic interpretations and can be efficient for block-wise small to medium scale problems. The Benders algorithm was also generalized to allow the solution of separable nonlinear problems (Molina, 1979). Pigot (1964) published a "double decomposition method" that could be thought as a combination of the Dantzig-Wolfe and Benders approaches to give upper and lower bounds of the optimal solution, but its implementation is complicated. After the publication of article "Two Level Planning" (Kornai and Liptak, 1965), more interest has developed in the decomposition of a mathematical programming problem. For example, Sanders (1965) published a method of primal decomposition of nonlinear programming problems by means of the Lagrange multipliers. Recently, an auction-based/price-driven coordination method was developed by Jose and Ungar (1998*a*; 1998*b*) on the basis of the work of Jennergren (1973). This method can handle situations in which the Lagrangian-based method may be inadequate, by using so called "slack resource" auction.

---

[1] Part of this chapter has been published in Cheng *et al.* (2005*b*; 2006*b*)

Because of the elegance of the economic interpretation of the Dantzig-Wolfe decomposition, and the more recent price-driven coordination method in plant-wide optimization, this chapter concentrates its discussion on two methods: an extension of the Dantzig-Wolfe decomposition method (Whinston, 1966) and a price-driven coordination method.

3.1.2 Extension of Dantzig-Wolfe Decomposition

The Dantzig-Wolfe decomposition for solving large-scale LP problems can be naturally extended to solving large-scale QP problems, as QP problems can be converted into a form that can be solved by the Simplex method (Panne, 1975). By applying the optimality conditions, a QP problem can be transformed to a linear complementarity problem. The linear complementarity problem consists of a system of linear equations and complementarity equations. In the Simplex method, the nonlinear complementarity equations can be handled by a complementary pivoting rule.

The extension of Danzig-Wolfe decomposition to solve large-scale QP problem is described in Whinston (1966). This decomposition algorithm arises naturally where the coefficient matrix of the constraints has a block-wise structure, and thus a large problem can be decomposed into a collection of smaller subproblems. These subproblems are then coordinated by using linking constraints. To illustrate the idea of complementary pivoting for the extension of Dantzig-Wolfe decomposition, a QP problem with two subsystems is considered as follows:

$$\max_{\mathbf{x1},\mathbf{x2}} f(\mathbf{x}_1,\mathbf{x}_2) = \mathbf{c}_1^T\mathbf{x}_1 + \mathbf{c}_2^T\mathbf{x}_2 - \frac{1}{2}\left[\mathbf{x}_1^T\ \mathbf{x}_2^T\right]\mathbf{Q}\begin{bmatrix}\mathbf{x}_1\\ \mathbf{x}_2\end{bmatrix}$$

$$\mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 = \mathbf{b}_0 \tag{3.1}$$

$$\mathbf{B}_1\mathbf{x}_1 \leq \mathbf{b}_1 \tag{3.2}$$

$$\mathbf{B}_2\mathbf{x}_2 \leq \mathbf{b}_2 \tag{3.3}$$

$$\mathbf{x}_1 \geq 0 \quad \mathbf{x}_2 \geq 0 \quad \mathbf{Q} = \begin{bmatrix}\mathbf{Q}_{11} & \mathbf{Q}_{12}\\ \mathbf{Q}_{21} & \mathbf{Q}_{22}\end{bmatrix}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are two vectors of decision variables, $\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2$ are the corresponding RHS of the linking constraints and the subproblem constraints respectively. $\mathbf{c}_1$ and $\mathbf{c}_2$ are cost coefficients in the objective function associated with decision variables $\mathbf{x}_1$ and $\mathbf{x}_2$, and matrix $\mathbf{Q}$ is a symmetric positive semidefinite matrix. The convex sets described by the inequalities (3.2) and (3.3) are assumed to be bounded to simplify the discussion [2].

The above QP formulation can be converted into its equivalent master problem (MP) by using the theorem of convex combination (Lasdon, 2002). Therefore, using the convex combination in equation (2.5) and (2.6) in Chapter 2 and substituting them into inequality (3.2) and (3.3) of the original problem, the following master problem can be obtained:

$$\begin{aligned}\max_{\lambda_i,\beta_j} F(\lambda_i,\beta_j) = \mathbf{c}_1^T\sum\lambda_i\mathbf{x}_{i1} + \mathbf{c}_2^T\sum\beta_j\mathbf{x}_{j2} - \frac{1}{2}\sum\lambda_i\mathbf{x}_{i1}^T\mathbf{Q}_{11}\sum\lambda_i\mathbf{x}_{i1}\\ -\sum\lambda_i\mathbf{x}_{i1}^T\mathbf{Q}_{12}\sum\beta_j\mathbf{x}_{j2} - \frac{1}{2}\sum\beta_j\mathbf{x}_{j2}^T\mathbf{Q}_{22}\sum\beta_j\mathbf{x}_{j2}\end{aligned}$$

$$\mathbf{A}_1\sum\lambda_i\mathbf{x}_{i1} + \mathbf{A}_2\sum\beta_j\mathbf{x}_{j2} = \mathbf{b}_0 \tag{3.4}$$

$$\sum\lambda_i = 1 \quad \sum\beta_j = 1 \tag{3.5}$$

$$\lambda_i \geq 0 \quad \beta_j \geq 0$$

where $\mathbf{x}_{i1}$ and $\mathbf{x}_{j2}$ represent the set of extreme points of the feasible region for subproblems 1 and 2 respectively, and λ_i and β_j are the non-negative coefficients in the convex combination. The transformed master problem is still a concave programming problem which has been proved in Whinston (1966).

---

[2] Without this assumption, one can follow the argument developed by Dantzig and Wolfe in (1960)

The master problem can be converted into a system of linear complementarity equations by applying the optimality conditions. The only nonlinear constraints are shown in equation (3.6):

$$u_i \lambda_i = 0 \quad w_j \beta_j = 0 \quad \text{for all i, j} \tag{3.6}$$

where u_i and w_j are the Lagrange multipliers for the non-negativity constraints for λ_i and β_j. The above linear complementarity problem can be solved using the Simplex method with some modifications. At the optimum, the complementarity constraints have to be satisfied. In the Simplex tableau, either u_i and w_j or λ_i and β_j are in the basic set. Through a complementarity pivoting rule, the variables entering and leaving the basic set are chosen to make sure that the complementarity constraints are satisfied.

The extension of Dantzig-Wolfe decomposition, which incorporates the complementary pivoting rule for solving large-scale QP problems, was discussed in Whinston (1966). The procedure can be briefly described as follows:

Step 1. Determine the most negative u_i or w_j variable by solving the subproblems, and the algorithm will terminate when no negative variable can be found.

Step 2. Introduce into the basis the corresponding complementary variable λ_i or β_j of the priced-out variable determined in Step 1. When Simplex pivoting is performed, the leaving variable should be chosen from the λ_i or β_j in the basis and the variable u_i or w_j designated in Step 1. If the variable designated in Step 1 is removed, return to Step 1; otherwise, go on to Step 3.

Step 3. Introduce the u_i or w_j variable into the basis that is complementary to the leaving variable in Step 2. Carry out another Simplex pivoting. The leaving variable is then chosen from the λ_i or β_j in the basis and the variable designated in Step 1. If a λ_i or β_j is chosen to be leaving, repeat Step 3; otherwise, go back to Step 1.

The major steps in solving large-scale LP and QP using Dantzig-Wolfe decomposition are almost the same except those described above. In the above algorithm, we are either introducing a λ_i^* or β_j^* variable complementary to u_i^* or w_j^* designated by Step 1 or some u_i or w_j variable. Since we can determine the tableau elements of the particular generated column in the basis at any iteration, all the variables in the basis can be determined throughout the algorithm.

3.1.3 Discussions

For the extension of Dantzig-Wolfe decomposition algorithm discussed in the previous section, it is observed that the dimension of the restricted master problem (RMP) may increase as the coordination process continues, and to our knowledge, no attempt has been found to maintain a fixed RMP size as in the Dantzig-Wolfe decomposition for LP problems. Based on some preliminary computational studies, Figure 3.1 illustrates the worst case of this extension of Dantzig-Wolfe decomposition for QP problems, where the dimension of RMP can contain as many as the number of total extreme points of subproblems. For real-time solution of large-scale QP problems, this can be a serious barrier to a practical implementation of this extension.

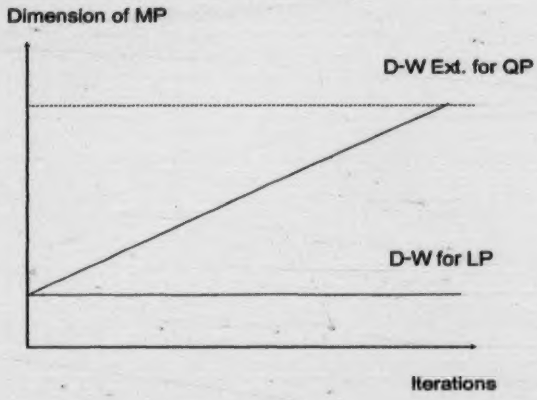


Figure 3.1: Limitations of D-W decomposition for QP

Because of the limitation of the extension of Dantzig-Wolfe decomposition, we turn to auction-based (or price-driven) coordination method for solving large-scale QP problems with the described separable structure.

3.1.4 Price-driven Coordination Methods

The price-driven coordination method discussed in Jose and Ungar (1998*a*; 1998*b*) can be used to solve resource distribution or auction problems. In their method, a large-scale optimization problem is decomposed into subproblems by relaxing the resource constraints which connect the subsystems together (the linking constraints in our previous discussions). The general large-scale nonlinear optimization problem considered here is:

$$\max_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \sum_i^n \mathbf{f}_i(\mathbf{x}_i)$$

subject to:

$$\sum_i^n R_i(\mathbf{x}_i) \leq \overline{R} \tag{3.7}$$
$$\mathbf{x}_i \in X_i$$

where $\mathbf{x}_i$ is an n_i-vector of decision variables, X_i is the feasible solution set of the i^{th} subproblem, $\mathbf{f}_i$ is the objective function for subproblem i, R_i is an m-vector of resource demands for subsystem i, and $\overline{R}$ represents the availability of common resources. The subproblems for each unit are re-formulated as follows:

$$\max_{\mathbf{x}_i \in X_i} \quad \mathbf{f}_i(\mathbf{x}_i) - (\mathbf{p} + qR_i(\mathbf{x}_i))^T R_i(\mathbf{x}_i) \tag{3.8}$$

where $\mathbf{p} \in R_+^m$ is a given price vector and q is a small positive scalar[3]. It was shown in Jose and Ungar (1998*b*) that if the subproblems have concave, continuous objective functions and compact, convex feasible sets, there exist equilibrium augmented prices in the form of $\mathbf{p} + qR$ that optimally coordinate the subproblems for given resource availability $\overline{R}$. For a given q, the equilibrium prices satisfy the nonlinear complementarity problem (NCP):

$$\begin{aligned} \Delta(\mathbf{p}, q) = \sum_i R_i(\mathbf{p}, q) - \overline{R} &\leq \mathbf{0} \\ \mathbf{p}^T(\Delta(\mathbf{p}, q)) &= 0 \\ \mathbf{p} &\geq \mathbf{0} \end{aligned} \tag{3.9}$$

---

[3] For example, the small positive scalar q is chosen as 0.01 in Jose and Ungar (1998*b*) for solving linear programs (as a special case), and can be set to 0 for solving nonlinear programs (including quadratic programs). More detailed discussion can be found in Jose and Ungar (1998*b*).

where $\mathbf{\Delta}(\mathbf{p}, q) \in R^m$ is the corresponding *excess resource demand* (i.e., the difference between the total demand of all subproblems and the plant-wide resource availability). The optimum of problem (3.7) can be obtained by solving the subproblems as given by (3.8) independently, given the equilibrium prices from problem (3.9). The mechanism for adjusting p until it satisfies problem (3.9) can be considered as the coordination in price-driven approach.

3.2 An Efficient Price Update Scheme

In many applications, the linking constraints take the form of (separable) equality constraints[4]. For example, the availability of (common) strategic resources, such as gasoline and ammunition, are the linking constraints and usually limited and scarce in most military resource allocation problems. When the linking constraints are equality constraints, which are frequently encountered in applications, the NCP in Equation (3.9) is simplified to:

$$\mathbf{\Delta}(\mathbf{p}, q) = \mathbf{0} \tag{3.10}$$

Therefore, the price vector can be adjusted by numerically solving a system of equations.

3.2.1 Newton's Method

For well-posed problems of the form given in (3.10), Newton's method can be used. Then during an iteration, the coordinator adjusts prices as follows:

$$\mathbf{p}(k+1) = \mathbf{p}(k) - \alpha \mathbf{J}^{-1}\mathbf{\Delta}(k) \tag{3.11}$$

where $\mathbf{\Delta} \triangleq \mathbf{\Delta}(\mathbf{p}, q)$ for simplicity, α is the step size, and

$$\mathbf{J} = \frac{d\mathbf{\Delta}(k)}{d\mathbf{p}(k)} = \sum_i \frac{dR_i(k)}{d\mathbf{p}(k)} \tag{3.12}$$

---

[4] Otherwise, an NCP given in (3.9), which is generally more difficult, needs to be solved through NCP algorithms such as Non-smooth Newton Methods, Smoothing Methods, and Jacobian Smoothing Methods discussed in Kanzow and Pieter (1999)

assuming the matrix in (3.12) is invertible. For the purpose of explanation, here we define $\mathbf{J}_i$ as the component of the overall Jacobian matrix $\mathbf{J}$, which corresponds to subproblem i, i.e., $\mathbf{J} = \sum_i \mathbf{A}_i\mathbf{J}_i$. Figure 3.2 explains the major information exchange between the coordinator and subsystems.

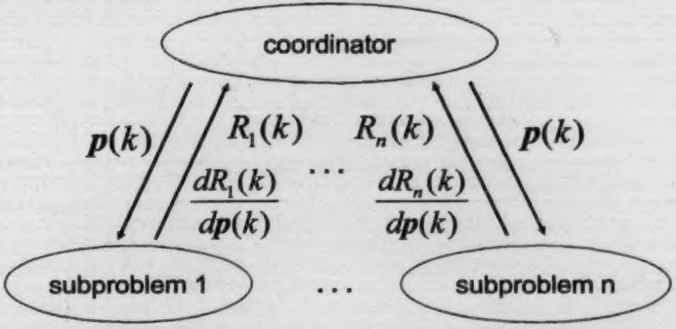


Figure 3.2: Infomation Flow in Price-driven Coordination

The coordination process is similar to setting up the prices for selling common resources to different consumers. The coordinator sends a price vector that contains pricing information for the resources to every subsystem. After solving their local optimization problems, the subsystems inform the coordinator of the resource demands (R_i) at current prices and their responses to the price change ($dR_i/d\mathbf{p}$). The coordinator then collects these two pieces of information to evaluate $\boldsymbol{\Delta}$ and $d\boldsymbol{\Delta}/d\mathbf{p}$, and the prices are updated using equation (3.11). This process of information exchange continues until the total demand is equal to overall supply, i.e., $\boldsymbol{\Delta} = 0$.

3.2.2 Jacobian Evaluation

The sensitivity $dR_i/d\mathbf{p}$ of the local subproblems can be obtained by standard post-optimality analysis techniques (McCormick, 1983; Wolbert *et al.*, 1994). In this work, the modified QP subproblems have the form:

$$\min_{\mathbf{x}_i} \quad (\mathbf{c}_i^T - \mathbf{p}^T\mathbf{A}_i)\mathbf{x}_i - \frac{1}{2}\mathbf{x}_i^T(\mathbf{Q}_i + q\mathbf{A}_i^T\mathbf{A}_i)\mathbf{x}_i$$

subject to:

$$\mathbf{B}_i^{eq}\mathbf{x}_i = \mathbf{b}_i^{eq} \tag{3.13}$$
$$\mathbf{B}_i^{ineq}\mathbf{x}_i \le \mathbf{b}_i^{ineq}$$

where $R_i(\mathbf{x}_i) = \mathbf{A}_i\mathbf{x}_i$, and

$$\sum_i \frac{dR_i(k)}{d\mathbf{p}(k)} = \sum_i \mathbf{A}_i \frac{d\mathbf{x}_i(k)}{d\mathbf{p}(k)}$$

where $\mathbf{A}_i$ is the coefficient matrix in the linking constraints corresponding to the variables of the i^{th} subproblem. Following the standard approach to sensitivity analysis, a Lagrangian function is firstly constructed for problem (3.13):

$$L_i(\mathbf{x}_i, \boldsymbol{\lambda}_i, \boldsymbol{\mu}_i, \mathbf{p}) = \bar{\mathbf{f}}_i^T\mathbf{x}_i - \frac{1}{2}\mathbf{x}_i^T\bar{\mathbf{H}}_i\mathbf{x}_i + \boldsymbol{\lambda}_i^T(\mathbf{B}_i^{eq}\mathbf{x}_i - \mathbf{b}_i^{eq}) - \boldsymbol{\mu}_i^T(\mathbf{B}_i^{ineq}\mathbf{x}_i - \mathbf{b}_i^{ineq}) \tag{3.14}$$

where

$$\bar{\mathbf{f}}_i^T = \mathbf{c}_i^T - \mathbf{p}^T\mathbf{A}_i \quad \text{and} \quad \bar{\mathbf{H}}_i = \mathbf{Q}_i + q\mathbf{A}_i^T\mathbf{A}_i \tag{3.15}$$

The first-order optimality conditions are:

$$\begin{aligned} \nabla_{\mathbf{x}_i} L_i(\mathbf{x}_i, \boldsymbol{\lambda}_{i,A}\,\boldsymbol{\mu}_i, \mathbf{p}) = \bar{\mathbf{f}}_i - \bar{\mathbf{H}}_i\mathbf{x}_i + (\mathbf{B}_i^{eq})^T\boldsymbol{\lambda}_i + ({}_A\mathbf{B}_i^{ineq})_A^T\boldsymbol{\mu}_i &= \mathbf{0} \\ F_i(\mathbf{x}_i, \mathbf{p}) = \mathbf{B}_i^{eq}\mathbf{x}_i - \mathbf{b}_i^{eq} &= \mathbf{0} \\ {}_A\mathbf{g}_i(\mathbf{x}_i, \mathbf{p}) = {}_A\mathbf{B}_i^{ineq}\mathbf{x}_i - {}_A\mathbf{b}_i^{ineq} &= \mathbf{0} \\ {}_I\mathbf{g}_i(\mathbf{x}_i, \mathbf{p}) + {}_I\boldsymbol{\sigma}_i = {}_I\mathbf{B}_i^{ineq}\mathbf{x}_i - {}_I\mathbf{b}_i^{ineq} + {}_I\boldsymbol{\sigma}_i &= \mathbf{0} \end{aligned} \tag{3.16}$$

The vectors $\boldsymbol{\lambda}$ and ${}_A\boldsymbol{\mu}$ are the Lagrange multipliers for the equality constraints and active inequality constraints in problem (3.13), respectively, and the vector ${}_I\boldsymbol{\sigma}$ is the slack variables corresponding to the inactive inequality constraints. The subscripts A and I indicate the active and inactive status of the inequality constraints.

The sensitivity matrices can be obtained by differentiating the optimality conditions in (3.16). One may notice that, we have extended the sensitivity analysis derivation in Wolbert *et al.* (1994) by including the sensitivity of slack variables of the inactive inequality constraints. Then, we obtain a system of equations as follows:

$$
\begin{aligned}
\nabla^2_{\mathbf{x}_i\mathbf{x}_i} L_i d\mathbf{x}_i + \nabla^2_{\mathbf{x}_i\mathbf{p}} L_i d\mathbf{p} + \nabla_{\mathbf{x}_i} F_i^T d\boldsymbol{\lambda}_i + \nabla_{\mathbf{x}_i A}\mathbf{g}_i{}^T d_A \boldsymbol{\mu}_i &= \mathbf{0} \\
\nabla_{\mathbf{x}_i} F_i d\mathbf{x}_i + \nabla_{\mathbf{p}} F_i d\mathbf{p} &= \mathbf{0} \\
\nabla_{\mathbf{x}_i A}\mathbf{g}_i d\mathbf{x}_i + \nabla_{\mathbf{p}A}\mathbf{g}_i d\mathbf{p} &= \mathbf{0} \\
\nabla_{\mathbf{x}_i I}\mathbf{g}_i d\mathbf{x}_i + \nabla_{\mathbf{p}I}\mathbf{g}_i d\mathbf{p} + d_I \boldsymbol{\sigma}_i &= \mathbf{0}
\end{aligned}
\tag{3.17}
$$

For QP problems, the derivatives in (3.17) are constant matrices. Then the sensitivity matrix, $d\mathbf{x}_i/d\mathbf{p}$, can be obtained by solving the following system of linear equations assuming that the matrix $\boldsymbol{\Gamma}_i$ is full rank:

$$
\boldsymbol{\Gamma}_i \begin{bmatrix} \nabla_{\mathbf{p}}\mathbf{x}_i \\ \nabla_{\mathbf{p}}\boldsymbol{\lambda}_i \\ \nabla_{\mathbf{p}A}\boldsymbol{\mu}_i \\ \nabla_{\mathbf{p}I}\boldsymbol{\sigma}_i \end{bmatrix} = - \begin{bmatrix} \mathbf{A}_i^T \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}
\tag{3.18}
$$

where

$$
\boldsymbol{\Gamma}_i = \begin{bmatrix} \mathbf{Q}_i + q\mathbf{A}_i^T\mathbf{A}_i & \mathbf{B}_i^{eqT} & {}_A\mathbf{B}_i^{ineqT} & \mathbf{0} \\ \mathbf{B}_i^{eq} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ {}_A\mathbf{B}_i^{ineq} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ {}_I\mathbf{B}_i^{ineq} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}
\tag{3.19}
$$

By solving the equations in (3.18), $\mathbf{J}_i = \frac{d\mathbf{x}_i}{d\mathbf{p}}$ can be obtained straightforwardly. Therefore, the overall Jacobian matrix $\mathbf{J} = \sum_i \mathbf{A}_i \mathbf{J}_i$ can be calculated for updating the price vector $\mathbf{p}$ in equation (3.11).

3.2.3 Step Size Determination

The Jacobian matrix in (3.12) is valid only when there is no active set change; however, there is no guarantee that the active set for any subproblem will not change during a solution. Therefore, a full Newton step is taken only when no active set change in each subproblem is identified. Otherwise, a step size α, less than one, should be taken.

The largest step size that could be taken before a change in active set occurs can be determined from the sensitivity information. A trust region specification is performed to identify the smallest step size α that causes a change to current active set in each subproblem. When there is an active set change, one of the slack variables and Lagrange multipliers in the subproblems will become zero. The slack variables or Lagrange multipliers, denoted by $\boldsymbol{\theta}$, as a function of p is given as follows:

$$\boldsymbol{\theta} = \boldsymbol{\theta}^*(\mathbf{p}(k)) + \nabla_{\mathbf{p}}\boldsymbol{\theta}(\mathbf{p} - \mathbf{p}(k)) \tag{3.20}$$

Then equation (3.11) can be substituted into equation (3.20) to express $\boldsymbol{\theta}$ in terms of α. We can equate $\boldsymbol{\theta}$ to 0 for every slack variable and Lagrange multiplier for each subproblem to determine the value of α which makes individual constraint change its activity[5]. The smallest positive α will be taken as the step size candidate for the current iteration. If it is less than 1, it will be chosen as the step size; otherwise, a full Newton's step is taken. Although this procedure is not shown in Figure 3.2, the above calculation can be implemented in the subproblems as α can be determined independently for each subproblem.

3.2.4 Algorithmic Statement

There are two phases in one coordination iteration. Each phase contains an information distribution-gathering procedure, in which information exchange happens between the coordinator and the subsystems. The price-adjustment algorithm is summarized as follows:

- Step 1 (phase 1): Trust region specification determines a nonnegative step size to update the prices in equation (3.11). Then the updated prices are distributed to subproblems. For initialization, the prices could be set to zero or some other initial guesses.

- Step 2 (phase 1): Each subproblem is independently solved based on current price information, and its solution $\mathbf{x}_i$ and sensitivity information $\nabla_{\mathbf{p}}\boldsymbol{\lambda}_i$, $\nabla_{\mathbf{p}}\boldsymbol{\mu}_i$, and $\nabla_{\mathbf{p}}\boldsymbol{\sigma}_i$ are submitted to the coordinator.

---

[5] We assume that no degeneracy happens at the optimum, otherwise, the user should re-formulate the problem.

- Step 3 (phase 2): The coordinator calculates the excess demand $\boldsymbol{\Delta}$ and determines $\boldsymbol{\theta}$ using equation (3.18) and (3.12). Then the plant-wide Newton direction $\mathbf{J}^{-1}\boldsymbol{\Delta}$ is distributed to the subproblems. The whole procedure is terminated when the excess demand reaches zero.

- Step 4 (phase 2): By combining the sensitivity information recorded in step 2 and the plant-wide Newton direction, an active set change identification is performed independently for each subproblem. Then a set of allowable step size $\{\alpha_i\}$ is submitted to the coordinator.

With the proposed price-update scheme, the auction-based coordination method may exhibit computational efficiency in solving a class of QP problems with block-wise structure in the *local* constraints, separable objective function, and linking equality constraints.

3.2.5 Discussions

To illustrate the improvement that we have made to the price-driven coordination method, in this discussion, the proposed price-update scheme is briefly compared with the P-control price-update scheme discussed in Jose and Ungar (1998*b*).

In the P-control approach, the price vector is updated by:

$$\mathbf{p}(k+1) = \mathbf{p}(k) + k_c\boldsymbol{\Delta}(k) \tag{3.21}$$

Consider the following QP problem:

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^T\mathbf{Q}\mathbf{x} - \mathbf{c}^T\mathbf{x}$$

subject to:

$$\begin{aligned} \mathbf{A}\mathbf{x} &\leq \mathbf{b} \\ \mathbf{x} &\geq \mathbf{0} \end{aligned} \tag{3.22}$$

where

$$\mathbf{A} = \begin{bmatrix} 2 & 5 & 7 & 3 \\ 3 & 5 & 3 & 4 \\ 1 & 3 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 0 & 0 & 1.5 & 4 \\ 0 & 0 & 2 & 1 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 14 \\ 10 \\ 6 \\ 5 \\ 12 \\ 6 \end{bmatrix} \tag{3.23}$$

$$\mathbf{Q} = diag\{2,4,3,8\} \qquad \mathbf{c}^T = [2\ 5\ 6\ 8]$$

The problem can be decomposed into two subproblems with two linking constraints.

Table 3.1: Performance of different price-update strategies

Methods	Tuning	Iterations	Convergence
Newton's	NA	2	Yes
P-control	$k_c = 0.02$	400	Yes
P-control	$k_c = 0.04$	139	Yes
P-control	$k_c = 0.1$	NA	No

In Table 3.1, performance of these two algorithms is compared based on the number of iterations required to reach the equilibrium prices. Identical termination criteria and the initial guess $\mathbf{p} = \mathbf{0}$ are used in both algorithms. For this example, Figure 3.3 gives us a clear idea of the enhancement that has been made by applying the proposed price-adjustment algorithm. By using Newton's method, the price adjustment algorithm drives the solution to the optimum very fast, and balances the supply and demand efficiently. It should also be noted that the P-control price update scheme for the given controller tunings yields an oscillatory trajectory when we look at the objective function value of the overall problem. All evidence shows that the Newton-based price-adjustment algorithm provides a substantially faster convergence.

Furthermore, the proposed algorithm also provides a guideline for tuning the proportional gain in P-control scheme. Comparing equation (3.11) and (3.21) shows that

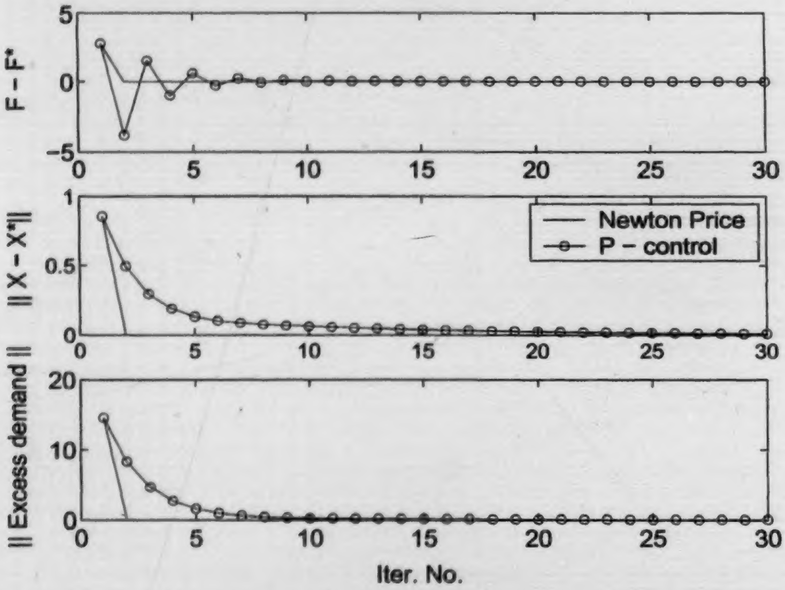


Figure 3.3: Convergence of different price update strategies

the proposed price-adjustment scheme adaptively updates the P-controller gain within each coordination cycle by setting $k_c = -\alpha \mathbf{J}^{-1}$. Moreover, for multi-variable systems (i.e., multiple resources), k_c is a diagonal matrix in the P-control scheme, while it is a full matrix in the proposed scheme.

3.3 Economic Interpretation of Coordination Mechanism

Microeconomics provides an interesting economic interpretation for the proposed coordination mechanism, and in particular, for the price adjustment scheme in equation (3.11). In this section, the single common resource case is used for the discussion (i.e., all the variables are scalars).

In a market, the price (p) of goods is related to the behavior of suppliers (or $S(p)$, quantity of goods supplied) and consumers (or, $D(p)$ the quantity of goods demanded. Standard microeconomics theory states that: as prices rise, supply will increase; as prices decrease, demand will increase. The equilibrium price is the price at which the demand

and supply are balanced.

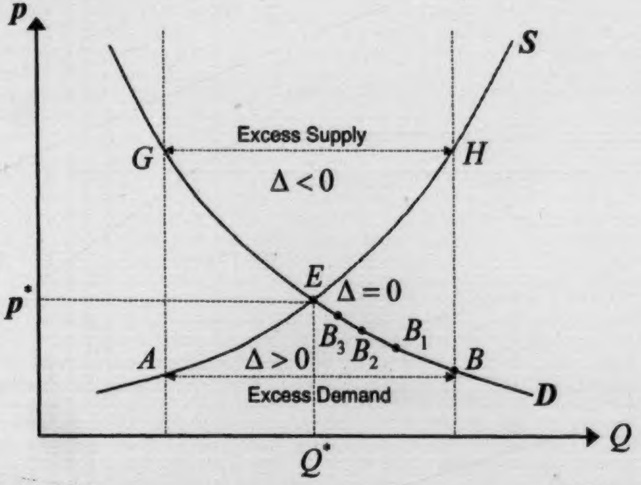


Figure 3.4: Determination of equilibrium price from supply and demand curves

These relationships are shown in Figure 3.4, where S is the supply curve and D is the demand curve. In an efficient market, supply and demand tend to be an equilibrium, which is denoted as point E on Figure 3.4. This point yields the equilibrium price p^*, which the coordinator is responsible for determing in the proposed coordination mechanism. Figure 3.4 also illustrates two important cases: excess supply (i.e., where $\Delta < 0$ and excess demand (i.e., where $\Delta > 0$). Then the value of Δ, as determined in equation (3.11), can be interpreted as one of these two cases, prior to convergence.

When $p > p^*$, excess supply tends to force price down; when $p < p^*$, excess demand tends to force price up. Consider starting at point B on Figure 3.4. This corresponds to an excess demand (i.e., more demand for resources than there is supply). If we were to increase the price of the resource, then demand should correspondingly decrease. As Δ is positive, the price update strategy given in equation (3.11) will increase prices, which will cause subproblems to decrease their demand for common resources in the next coordination cycle. This is shown by point B_1 in Figure 3.4. In every cycle of the coordination, the coordinator collects the information about the behavior of consumer, $R_i(p)$ (i.e., individual consumer demand) and $dR_i(p)/dp$ (i.e., response to price change or price elasticity of individual consumer's demand), at the current price p from every individual subproblem. Then, the coordinator calculates the overall demand $\sum_i R_i$ and elasticity $\sum_i \frac{dR_i}{dp}$, and uses them for price update. Subsequent coordination cycles drives the entire system along the demand curve toward the equilibrium price, as denoted by the points B_2, B_3 and so on. The techniques available in the literature and the proposed Newton-based methods for price updating follows this general strategy in slightly different ways.

3.4 Complexity Study

As was discussed in Chapter 2, the computational efficiency of a coordination strategy is a key factor in determining the viability of using coordinated decentralized optimization approaches in industrial applications. In this section, our interest is focused on investigating the scaling behavior of the price-driven coordination method via comprehensive computational studies, as well as a brief theoretical complexity analysis for the proposed price-adjustment scheme.

In this work, without loss of generality, consider a large-scale block-angular QP problem with p subproblems in form of:

$$\max \quad \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{f}^T\mathbf{x}$$

$$\text{s.t.} \quad \sum_i \mathbf{A}_i\mathbf{x}_i = \mathbf{b}_0 \tag{3.24}$$

$$\mathbf{B}_i\mathbf{x}_i \le \mathbf{b}_i \tag{3.25}$$

$$\mathbf{x}_i \ge \mathbf{0} \quad i = 1, 2, ..., p \tag{3.26}$$

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{Q}_1 & 0 & \dots & 0 \\ 0 & \mathbf{Q}_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \mathbf{Q}_p \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_p \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_p \end{bmatrix} \tag{3.27}$$

In the above, $\mathbf{x}_i$ is a vector of n_i decision variables. **H** is the Hessian matrix with block-diagonal matrices $\mathbf{Q}_i$, which have a dimension of $(n_i \times n_i)$ correspondingly; f is the cost coefficient of the linear term in the objective function, where $\mathbf{f}_i$ is a vector of n_i elements. $\mathbf{b}_i$ is the RHS of dimension m_i while $\mathbf{b}_0$ of m_0. The coefficient matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ have the corresponding dimension $(m_0 \times n_i)$ and $(m_i \times n_i)$, respectively.

In this work, the purpose of computational study is to determine the cost of solving a class of QP problems as described above, using price-driven coordination algorithms, where the term "cost" stands for the required arithmetic or other computational operations as is discussed before. Similarly, the concepts of "worst-case" behavior and "average-case" behavior (Nash and Sofer, 1996) are used to measure the cost of the algorithm.

3.4.1 Theoretical Analysis

Recall the coordination mechanism shown in Figure 3.2, the computational complexity of the price-driven coordination method has three contributors: the complexity of subproblems, the complexity of coordinator's calculation, and the coordination complexity which can be represented by communication cycle number (CCN). For discussions of the computational complexity of the decomposition algorithm, since the algorithms are investigated on the same sequential machine, the overall complexity can be expressed as:

$$\mathbf{T} = \{\mathbf{T}(CoProb) + \sum_{i=1}^{p} \mathbf{T}(SP_i)\} \times \mathbf{CCN} \tag{3.28}$$

Note that the complexity of the coordinator $\mathbf{T}(CoProb)$ is expressed differently from that in the Dantzig-Wolfe decomposition algorithm, where an RMP is referred to.

By following the similar convention to the complexity analysis for the Dantzig-Wolfe decomposition, we may have:

$$\mathbf{T}(NonCo) = \mathbf{T}(CoProb) + \sum_{i=1}^{p} \mathbf{T}(SP_i) \tag{3.29}$$

and the overall complexity can be expressed as:

$$\mathbf{T} = \mathbf{T}(NonCo) \times CCN \tag{3.30}$$

In theory, the choice of algorithms for solving subproblems and coordinator's problem (i.e., a system of equations) will not impact the coordination complexity (i.e., CCN), assuming different valid algorithms should be able to achieve identical[6] solutions for given inputs. Therefore, any state-of-the-art algorithms can be chosen to serve the purpose of complexity analysis. Next, the complexity is studied in three parts.

On Subproblems

The subproblems that we encounter when the price-driven coordination method is applied to solving large-scale QP problems are smaller scale QP problems. The *interior point methods* (IPM) and *active set methods* are common solution approaches for QP problems. To our knowledge, no polynomial time algorithm proof is given in the literature for active set methods. For the purpose of complexity studies, interior point methods are chosen as the solver since we can find abundant sources of complexity analysis of IPM for QP (Nash and Sofer, 1996; Potra and Wright, 2000; Illes and Terlaky, 2002).

The complexity study of interior point methods for QP problems itself is a difficult topic in the literature. The results are usually obtained and valid for some specified assumptions and implementations of algorithms. This work is not intended to derive an accurate complexity bound, but rather to identify the key factors that can lead to a polynomial-time or non-polynomial time price-driven coordination algorithm. We use the results from the literature to serve this purpose.

The interior point method uses Newton's method in the solution of a general QP problem, where each Newton iteration involves the solution of a set of equations, requiring $O(n^3)$ operations (Nash and Sofer, 1996) [7]. When the Hessian matrix Q in a QP problem is positive-semidefinite, with an appropriately chosen starting point, the IPM can converge to the solution in $O(n^2 \log \frac{\mu_0}{\epsilon})$ iterations (Potra and Wright, 2000), where μ_0 is the initial value of the barrier parameter and $\epsilon \leq 2^{-2L}$ is the tolerance to the solution for integer data length L.

---

[6] At least to some accuracy that numerically satisfies specified requirements.

[7] A better bound, $O(n^{2.5})$, for the cost of solving for Newton's iteration can be found on page 181 of (Illes and Terlaky, 2002)

Thus, it can be concluded that the interior point methods for solving convex QP problems are polynomial-time algorithms. In other words, the QP subproblems in the price-driven coordination method can be solved efficiently.

On the Coordinator Complexity

Recall that we have the formula (3.11) for the coordinator to update the price vectors. In the coordinator's computation, there are four major computational operations:

- With the information from the i^{th} subproblem, computing matrix inverse $\mathbf{\Gamma}_i^{-1}$;
- Obtaining $\frac{d\mathbf{x}_i}{d\mathbf{p}}$ by matrix-vector multiplication with the matrix inverse $\mathbf{\Gamma}_i^{-1}$ obtained beforehand;
- Calculating Jacobian matrix $\mathbf{J}$ by matrix multiplication $\mathbf{J}_i = \mathbf{A}_i \frac{d\mathbf{x}_i}{d\mathbf{p}}$ and the summation $\sum_{i=1}^{p} \mathbf{J}_i$;
- Computing the inverse of the overall Jacobian matrix $\mathbf{J}^{-1}$ and get the price vector updated.

It is known that matrix inversion takes more operations than matrix multiplication (including matrix-vector multiplication) and matrix summation. In the above four major operations, the matrix inverse $\mathbf{\Gamma}_i^{-1}$ is associated with the largest matrices that appear in the above four operations. Therefore, the matrix inversion $\mathbf{\Gamma}_i^{-1}$ contributes the dominant portion of the arithmetic operations. There are many matrix algorithms for calculating matrix inversion, for example, the Cholesky decomposition, the Householder reduction and the Winograd's method (Kronsjo, 1987). The upper bounds of the number of arithmetic operations are available for most of the methods, for instance, $\frac{n^3}{2} + 1.19n^{7/3} + n - 0.63n^{1/3}$ multiplications and $\frac{3n^3}{2} + 3.75n^{7/3} - 2n^2 + O(n^{5/3})$ additions are required for Winograd's method to get the inversion of an $n \times n$ matrix. To get a general expression, which also serves our purpose of complexity study, we can safely say that the matrix inversion can be completed in $O(n^3)$ arithmetic operations. It should be noted that, in the coordinator's computation, the matrix inversion $\mathbf{\Gamma}_i^{-1}$ and matrix-vector multiplication can be combined and calculated with LU or Cholesky decomposition methods. Although it does not change the conclusion from the complexity analysis, by avoiding explicit calculation of the matrix inverse, the required number of operations would be less and results in some improvement of computational performance.

In the price-driven coordination method, for a given QP problem in (3.24) to (3.26), $\mathbf{\Gamma}_i$ is an $(n_i + m_i) \times (n_i + m_i)$ matrix, thus it requires $O((n_i + m_i)^3)$ arithmetic operations to get its inversion. As long as the the number of subproblems p is finite and not very large[8], the operations required to calculate all the matrix inverse can be expressed as $O(p \times (\max(n_i + m_i)^3)$, which again requires polynomial operations with respect to the problem size.

It should be noted that, being different from Dantzig-Wolfe decomposition where the coordinator (RMP) has a fixed size and the coordinator complexity has nothing to do with the subproblem sizes, the complexity of coordinator's computation is related to the subproblem sizes.

On Coordination Complexity

The coordination complexity refers to the iterations of information exchanges between the coordinator and subproblems, i.e., it can be expressed by communication cycle number (CCN). To our knowledge, no theoretical results are available for the coordination complexity for the price-driven coordination method in the literature. One practical approach to investigating the coordination complexity is through comprehensive computational studies.

Before getting into the empirical study, let us qualitatively discuss the coordination complexity in the following two situations: the interior solution case (i.e., any subset $\mathbf{x}_i$ of the overall solution $\mathbf{x}$ lies in the interior of the feasible region of i^{th} subproblem) and the boundary solution case (i.e., at least one subset $\mathbf{x}_i$ of the overall solution $\mathbf{x}$ lies on the boundary of the feasible region of i^{th} subproblem). Recall that there is an active set change identification step in the price-driven coordination algorithm, which determines the Newton's step size.

---

[8] It is always possible to combine smaller problems into fewer bigger problems if we want to control the number of subproblems.

In the interior solution case, as long as the subset $\mathbf{x}_i^j$ at j^{th} communication cycle is in the interior of the feasible region of i^{th} subproblem, the subset $\mathbf{x}_i^{j+1}$ at $(j+1)^{th}$ communication cycle should be in the interior of the feasible region and a full Newton's step can be taken. Intuitively, since the subproblems are quadratic programs, the Newton's method can converge to the optimal solution in a few iterations, i.e., the CCN is small; however, in the case of boundary solution, whenever there is an active set change in the solution of any subproblem at j^{th} communication cycle, the overall Newton's step size α^j will be chosen from the Newton steps $\{\alpha^j | \min \alpha_i^j, \quad i = 1, ..., p\}$, and this will cause more iterations of information exchanges between the coordinator and subproblems.

In addition, common QP solvers, such as MATLAB® "quadprog", usually find the optimal solution in a few iterations when the optimal solution lies in the interior of the feasible region, i.e., no active inequality constraints at the optimum; but take more iterations to find the optimal solution when the optimum lies on the boundary of feasible region, i.e., there are bounding inequality constraints at the optimum. Therefore, it would be interesting to study the computational performance of the algorithms for the two cases, respectively.

3.4.2 Empirical Studies

For analysis of the computational complexity of the decomposition algorithm, since the algorithms are implemented on a sequential machine, we may express the overall complexity in terms of computation time:

$$\mathbf{t} = \sum^{CCN} \{\mathbf{t}(CoProb) + \sum_{i=1}^{p} \mathbf{t}(SP_i)\} \tag{3.31}$$

where t represents the total computation time to solve a problem. Through the comparison between the computation time of each subproblem and the total computational time, we can identify and focus on the "bottleneck" subproblem, which may be the largest in dimension or hardest to solve.

Similarly, if we denote the non-coordination computation time as:

$$\mathbf{t}(NonCo) = \mathbf{t}(CoProb) + \sum_{i=1}^{p} \mathbf{t}(SP_i) \tag{3.32}$$

and assume a distributed/parallel computing environment, for example, one CPU for each subproblem, then we have the equivalent computation time (parallel computing):

$$\mathbf{t}_{eqv}(NonCo) = \mathbf{t}(CoProb) + \max_{i=1}^{p}\{\mathbf{t}(SP_i)\} \tag{3.33}$$

As is discussed in Chapter 2, it may be desirable to balance the computational load on each computing node. Then, the computation time within a decentralized computing environment can be expressed as:

$$\mathbf{t}_{eqv} = \sum^{CCN} \mathbf{t}_{eqv}(NonCo) \tag{3.34}$$

Now the focus is on the analysis of communication cycle number CCN. Next, we are trying to determine the relationship between CCN and the characteristic parameters of the QP problems such as m_0, p and $|I_i| = (m_i \times n_i)$. Moreover, the influence of the relative subproblem ratio (RSR), which is defined as RSR $= \{\max\frac{|I_i|}{|I_j|},\ \ i,j = 1,2,...,p\}$, will also be studied, and this gives some idea on the computational load balance throughout the distributed computing network. It should be noted that, compared to the analysis for Dantzig-Wolfe decomposition for linear programming, the location of the optimum in the feasible region requires the investigation of the influence of the active constraints on the computational performance (especially the CCN) of the algorithm. This study may provide more insight into scaling issues for an optimization problem (e.g., the coordinated, decentralized MPC) when applying the proposed coordination algorithm.

In the following Monte Carlo simulations, besides the CCN, the computational efficiency and scaling behavior of the price-driven coordination method with the proposed price-update scheme will also be investigated. The performance of the coordination algorithm and the centralized QP solver will be compared within the MATLAB® platform. In both approaches, ILOG CPLEX 9.0 default Simplex solver is used to solve all QP problems and the coordinator's problem is solved by the proposed Newton's price-update method. In addition, for the purpose of comparison, we also reported the performance of CPLEX 9.0 interior point method (IPM) solver as a centralized QP solver. In this study, the focus is on the average behavior of different optimization strategies in solving the problems with the following assumptions:

1. The studied problems have bounded feasible regions and optimal solutions.
2. The problems are reasonably well scaled and no significant numerical errors will be encountered.

Similarly, due to the random nature of Monte Carlo simulation, we would like to acknowledge that there is possibility that an "average-case" problem may not be defined in our stochastic model.

The scheme for generating test problem instances is introduced in Appendix A.2.1. It randomly generates a class of QP problem instances with block-angular structure, as is given in equations (3.24) to (3.26). We start from a reference problem model, whose problem size and structure should be a good reference for the comparison experiments, i.e., we can observe the algorithm performance changes when we change the problem size and structure with respect to the reference model. With some preliminary tests, we choose the following set of parameters as the reference problem model:

$$\{p = 17, m_0 = 20, m_i = 15, n_i = 10 \quad i = 1, 2, ..., p\} \tag{3.35}$$

Note that the reference problem has subproblems of identical size, i.e., the problem is thus a "well-balanced" decomposable problem with $RSR = 1$. We also assume that each subproblem has been allocated to a distributed CPU. Therefore, the equivalent computational time $\mathbf{t}_{eqv}$ for the decomposition algorithm is estimated by summing up the time for solving the coordination problem and the most difficult subproblem, assuming a distributed computational environment. In the Monte Carlo simulation for each scenario, the number of problem instances is $200 \times 5 = 1000$, i.e., for each scenario, five runs of simulation are performed and each run solves 200 problem instances generated randomly.

Scenario 1: if we fix p and $|I_i|$, change m_0 (see Appendix A.2.2). In this case, we can study the performance of decomposition and coordination with respect to the dimension of linking constraints in equation (3.24).

For the interior solution case, Figure 3.5 shows that the CCN is fairly constant with respect to the number of linking constraints; while for the boundary solution case, Figure 3.6 shows that the CCN increases with increases in the dimension of linking constraints.

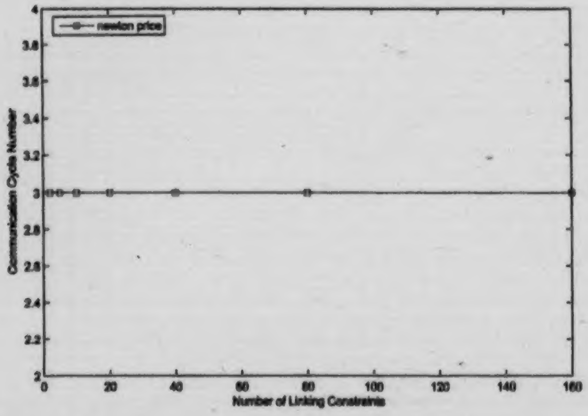


Figure 3.5: Interior case: CCN vs. number of linking constraints

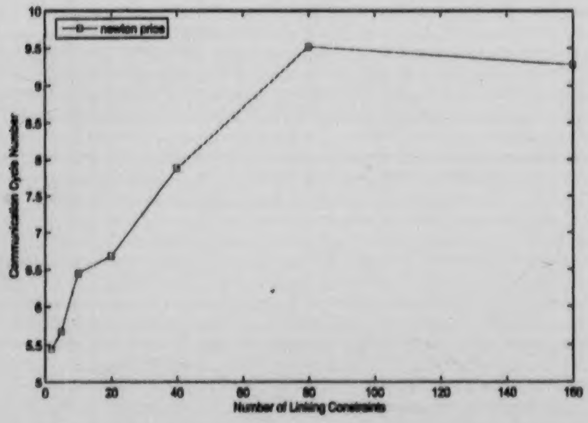


Figure 3.6: Boundary case: CCN vs. number of linking constraints

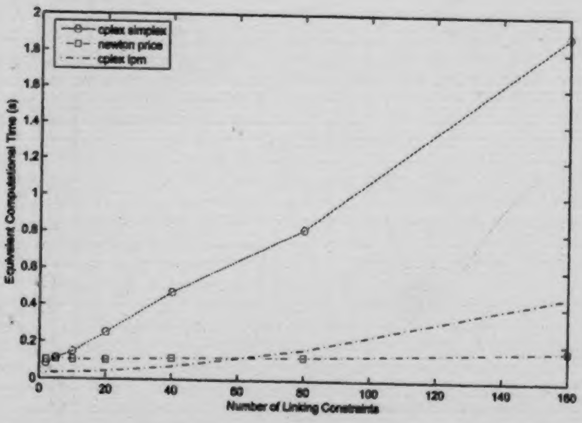


Figure 3.7: Interior case: computational performance vs. number of linking constraints

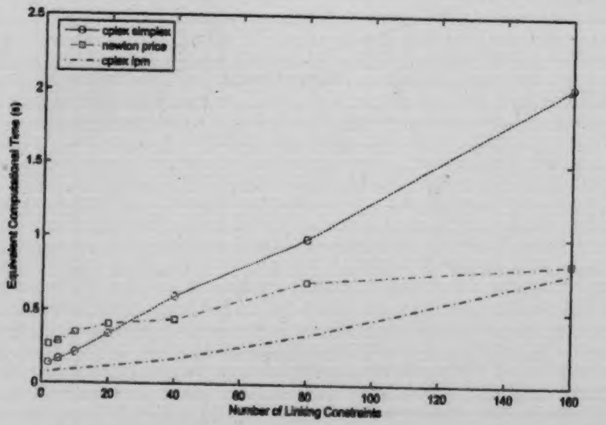


Figure 3.8: Boundary case: computational performance vs. number of linking constraints

When distributed computing is considered, Figure 3.7 shows that, in the case of interior solution, the price-driven coordination method always gives better computational performance than the CPLEX Simplex solver. Because the increase of the number of linking constraints does not impact the CCN in this case, the price-driven coordination algorithm even performs better than the CPLEX IPM solver. This implies the computational complexity of the coordination algorithm does not have strong dependence on the dimension of linking constraints. In the boundary solution case (cf. Figure 3.8), the coordination algorithm requires less computational time than CPLEX Simplex solver but more than the IPM solver when the number of linking constraints increases.

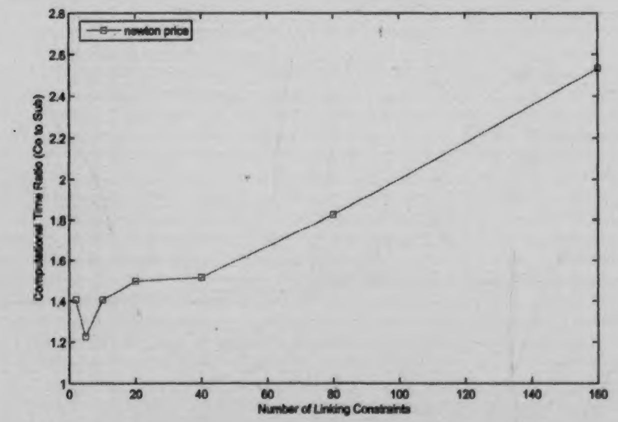


Figure 3.9: Interior case: computational time distribution vs. number of linking constraints

Figures 3.9 and 3.10 show the computational time ratio between the coordinator and the most "difficult" subproblem. It is evident that the computational effort to solve the coordinator problem increases, as the complexity of the coordinator problem increases as the number of linking constraints increases.

Scenario 2: For fixed p and m_0, we change subproblem size $|I_i|$ by simultaneously changing m_i and n_i (see Appendix A.2.2). In this case, we can study the algorithm performance with respect to subproblem sizes.

Figure 3.11 shows the coordination complexity in the interior solution case, which demonstrates showing that the size of subproblems does not have much impact on the

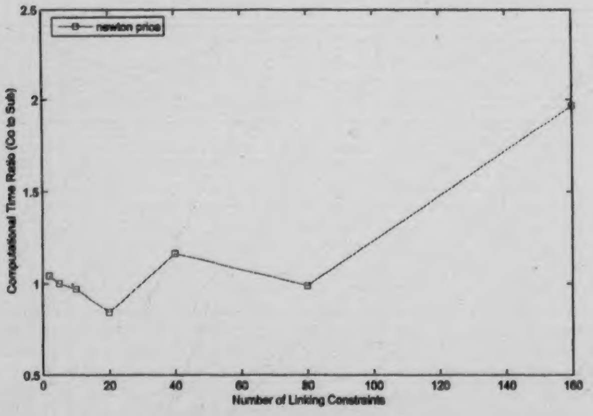


Figure 3.10: Boundary case: computational time distribution vs. number of linking constraints

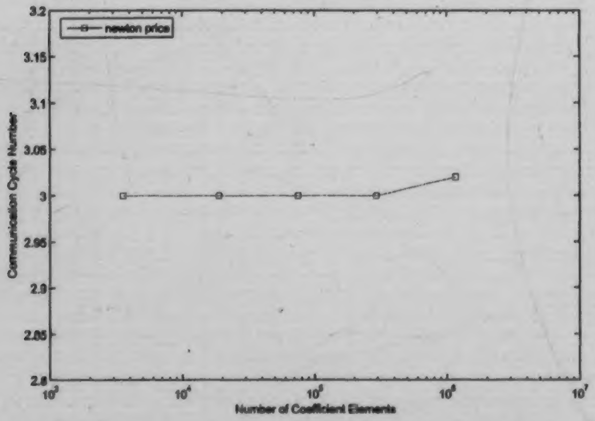


Figure 3.11: Interior case: CCN vs. subproblem size

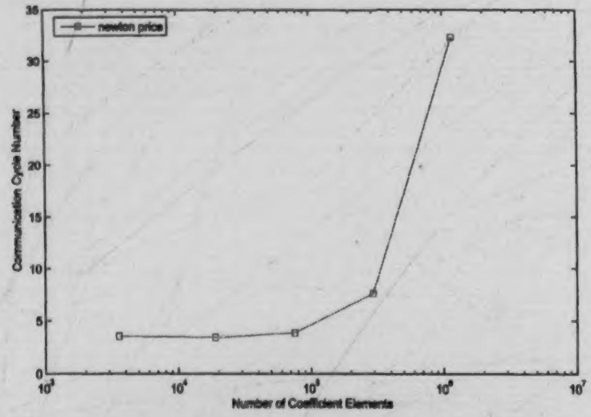


Figure 3.12: Boundary case: CCN vs. subproblem size

CCN; however, for the boundary solution case, Figure 3.12 shows that the CCN increases monotonically when the size of subproblems gets larger.

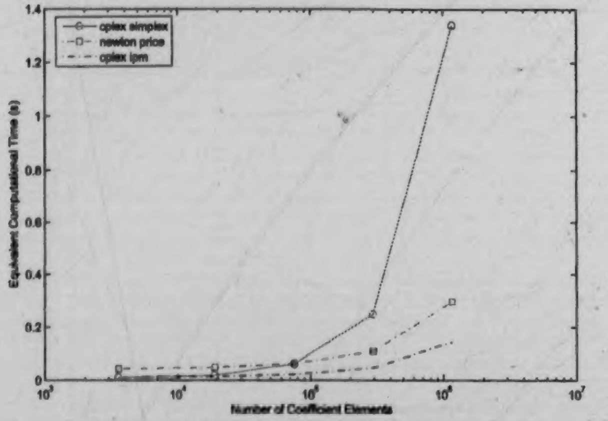


Figure 3.13: Interior case: computational performance vs. problem size

From Figure 3.13, it is also observed that the price-driven coordination method has higher computational efficiency than the centralized CPLEX Simplex solver when the overall problem size is large (i.e., when the subproblem size is larger than 15×10); however, Figure 3.14 shows the centralized CPLEX Simplex solver always outperforms the price-

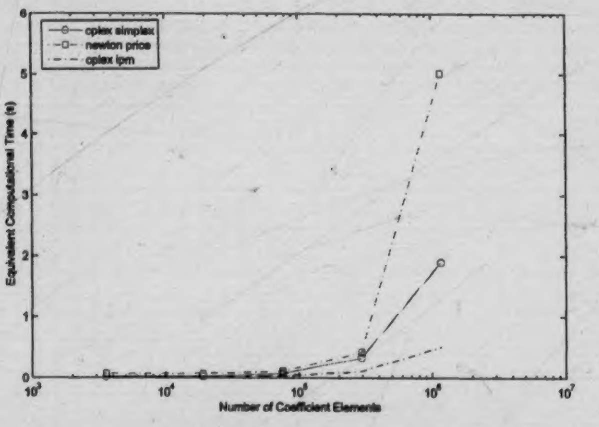


Figure 3.14: Boundary case: computational performance vs. problem size

driven coordination algorithm in the case of boundary solution. In addition, the centralized IPM solver shows higher efficiency in both cases.

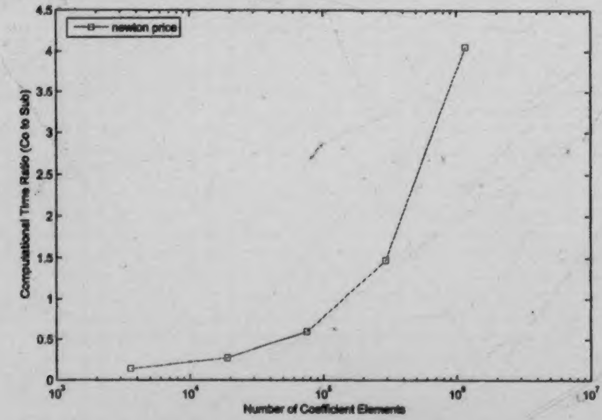


Figure 3.15: Interior case: computational time distribution vs. problem size

Again, Figure 3.15 and 3.16 show the computational time ratio between the coordinator and the most time consuming subproblem. Both figures show that the computational effort to solve the coordinator problem increases because the size of the coordinator problem increases as the subproblem size increases. But the increase of coordinator solution time

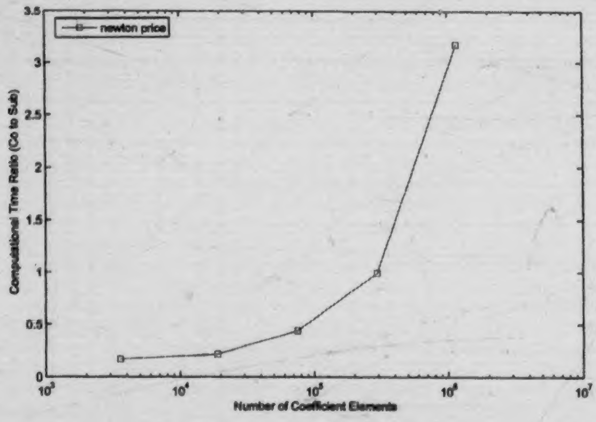


Figure 3.16: Boundary case: computational time distribution vs. problem size

in the interior solution case is more significant than that in the boundary solution case, which implies that the time for solving boundary case subproblems increases faster than the interior solution case.

Scenario 3: We keep m_0, m_i and n_i constant, and change the number of subproblems p (see Appendix A.2.2). In this case, we investigate the performance of the coordination algorithm when more and more subproblems are integrated into the coordination system, assuming a rather well-balanced subproblem computational load.

In Figure 3.17, the simulation results show that coordination complexity in the interior solution case is rather insensitive to the number of subproblems; however, for the boundary solution case, Figure 3.18 shows that the CCN increases when the number of subproblems increases.

Figure 3.19 and 3.20 show that the proposed price-driven coordination algorithm has higher computational efficiency than the centralized CPLEX Simplex solver in both cases. It should be noted that, in the interior solution case, the coordination algorithm can perform almost as well as the centralized IPM solver.

Both Figures 3.21 and 3.22 show that the computational effort to solve the coordinator problem becomes more dominant in the overall computational time when the number of subproblems increases.

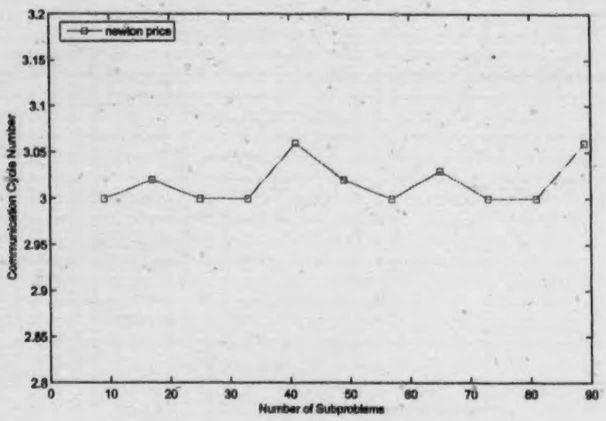


Figure 3.17: Interior case: CCN vs. number of subproblems

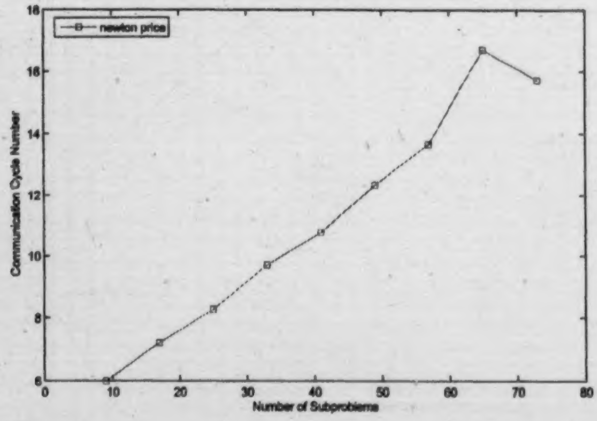


Figure 3.18: Boundary case: CCN vs. number of subproblems

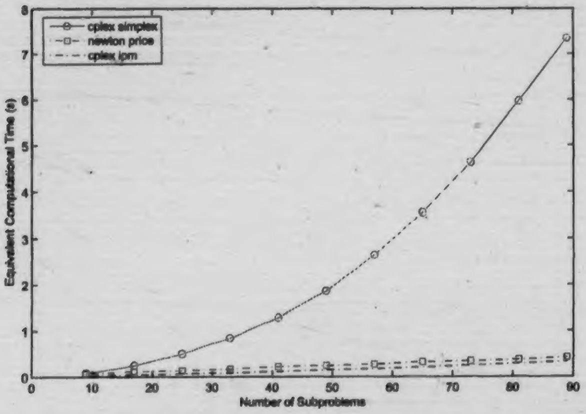


Figure 3.19: Interior case: computational performance vs. number of subproblems

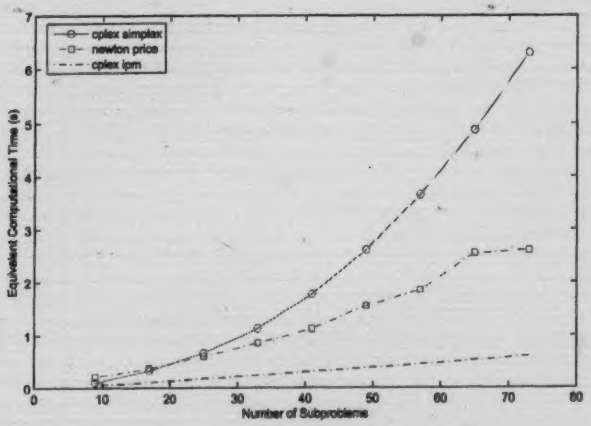


Figure 3.20: Boundary case: computational performance vs. number of subproblems

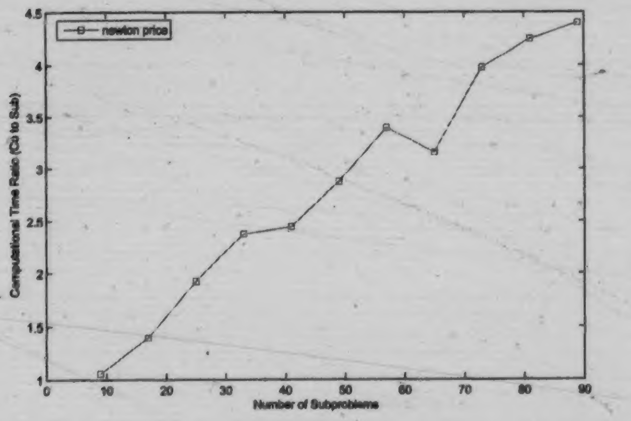


Figure 3.21: Interior case: computational time distribution vs. number of subproblems

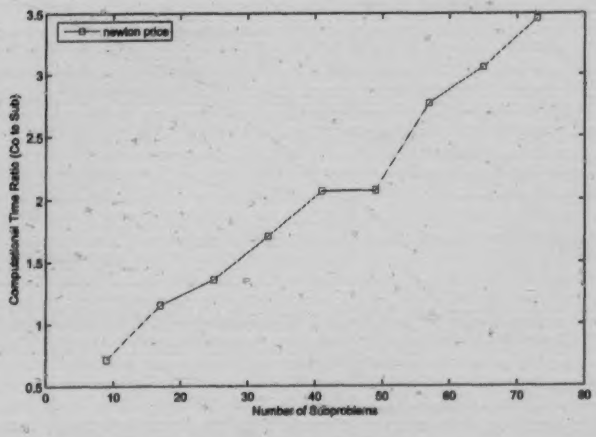


Figure 3.22: Boundary case: computational time distribution vs. number of subproblems

Scenario 4: If we fix m_0, $\sum_{i=1}^{p} m_i$ and $\sum_{i=1}^{p} n_i$, i.e., we fix the overall problem size, we can study the influence of relative subproblem ratio (RSR). In this case, we change p by combining subproblems into groups (see Appendix 2.2) according to different partition patterns of the original QP problem as we did for LP in Chapter 2. In the following studies, we arbitrarily choose one pattern for our investigation.

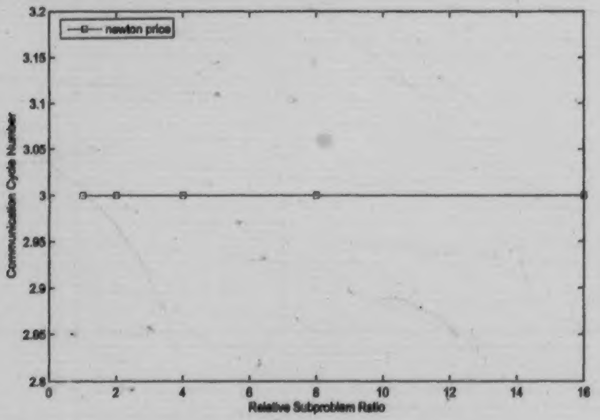


Figure 3.23: Interior case: CCN vs. RSR

For the interior solution case, Figure 3.23 shows that the imbalance of subproblem computational load (i.e., RSR) does not affect the coordination complexity; however, Figure 3.24 shows that the CCN increases in the boundary case when the imbalance of subproblems becomes more significant.

In Figure 3.25 and 3.26, we can see the proposed price-driven coordination algorithm has lower computational efficiency when the RSR increases in both cases, which implicates that the imbalance of subproblem load does affect the computational efficiency of the proposed algorithm.

The simulation results reported in Figure 3.27 and 3.28 shows that, although the overall problem size is the same in the computational studies, the coordinator's computational load increases when the imbalance of subproblem sizes gets more significant. This is because the complexity of the coordinator's problem is also determined by the largest subproblem as was discussed in previous sections.

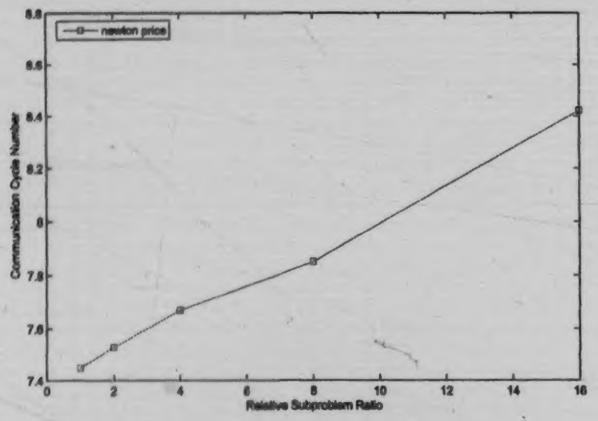


Figure 3.24: Boundary case: CCN vs. RSR

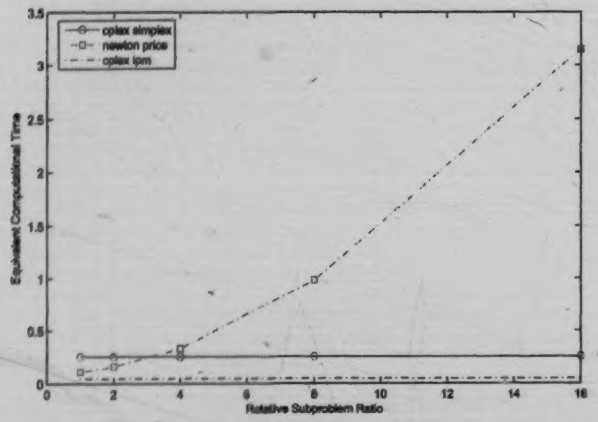


Figure 3.25: Interior case: computational performance vs. RSR

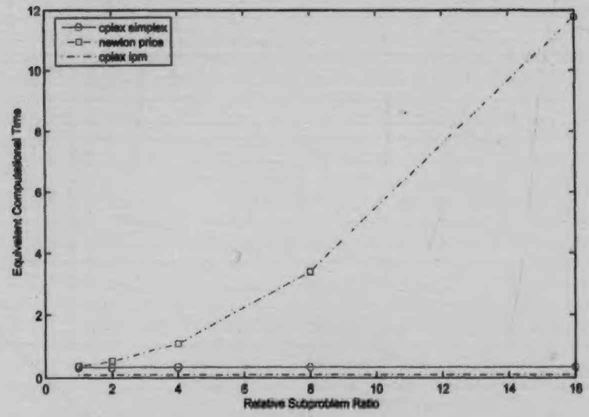


Figure 3.26: Boundary case: computational performance vs. RSR

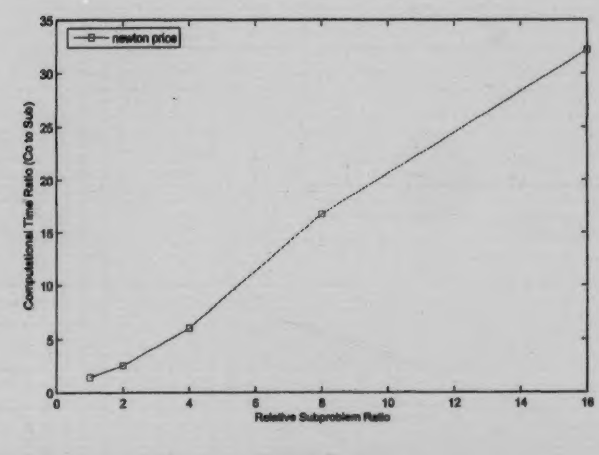


Figure 3.27: Interior case: computational time distribution vs. RSR

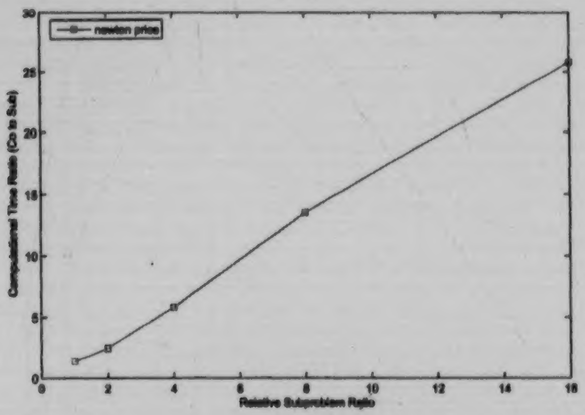


Figure 3.28: Boundary case: computational time distribution vs. RSR

Moreover, another interesting point may be about the relationship between the CCN and the number of active constraints for the overall problem in the boundary solution case. Recall our discussion on the coordination complexity in §3.4.1, more communication cycles are expected when there are more active constraints at the optimum solution. During the computational studies for the four scenarios, the number of active constraints was recorded, and thus is reported in the following figure.

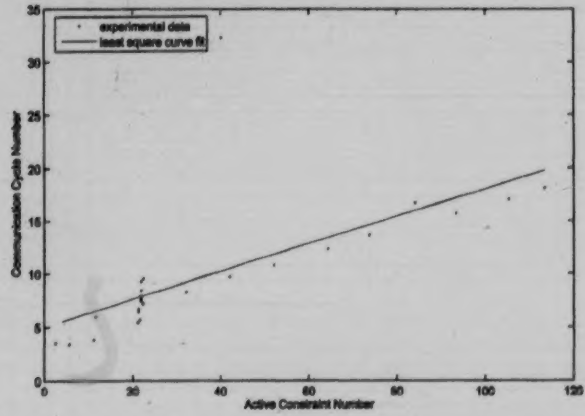


Figure 3.29: Boundary case: CCN vs. number of active constraints

It should be noted that the coordination complexity (CCN) increases almost linearly with the number of active constraints, which confirms our previous discussion.

Remarks: The complexity study performed in this chapter reveals some inherent features of the complexity of the price-driven coordination algorithm. From this study, we can draw some guidelines for coordination system design with the price-driven coordination method. In general, we may gain advantage by using the coordination algorithm to solve an QP problem, which has the following properties:

- having a block-angular structure;
- having high dimensional linking constraints;
- being assigned well-balanced computational load;
- having a large number of subproblems, if distributed computing power is available;
- having an interior solution at the optimum.

Furthermore, one observation from the simulation results is that the solution of the coordinator problem (i.e., price-update) contributes a dominant portion to the computational time for the overall problem. Thus, to speed up the overall coordination algorithm, a practical strategy can be the enhancement of the coordinator computing environment, such as a high-performance CPU or more efficient codes for solving equations and doing matrix operations.

3.5 Chapter Summary

In this chapter, decomposition and coordination strategies for solving large-scale quadratic programming problems have been investigated. In particular, an extension of Dantzig-Wolfe decomposition and the auction-based (or price-driven) coordination methods were studied. To improve the computational efficiency of the price-driven coordination method, an efficient price adjustment scheme was proposed by using Newton's method to incorporate the sensitivity information from subproblem solution. With the proposed price adjustment scheme, the computational performance of price-driven coordination methods is substantially enhanced when solving large-scale QP problems.

A well-designed structured complexity analysis approach was used to perform a thorough empirical study. New understanding of the relationships between computational performance and problem structural parameters was gained through a comprehensive study of the scaling behavior of price-driven coordination algorithm. The complexity study shows that, for large-scale quadratic programming problems with special structure, the proposed price-driven coordination algorithm can outperform centralized optimization solvers in some cases. The work in this chapter has made a significant step toward the application of price-driven coordination method for on-line solution of large-scale QP problems.

"The first requisite for success is the ability to apply your physical and mental energies to one problem incessantly without growing weary."

– by Thomas A. Edison

4

Coordinated, Decentralized MPC

In large-scale model predictive control (MPC) applications, such as plant-wide control and optimization, a large system is usually decomposed into several smaller subsystems and an individual controller is developed for each subsystem. This may lead to a decentralized or unit-based MPC framework. Such a control system may not be able to provide the plant-wide optimum operations because of its failure to consider the interactions between subsystems in decentralized MPC calculations. It has been identified that the coordination of the unit-based MPC systems can provide significant potential benefit. In this chapter, by applying the Dantzig-Wolfe decomposition and price-driven coordination methods, new approaches to coordinating decentralized MPC in the target calculation level are proposed. In the developed framework of designing a coordination system for decentralized MPC, only minor modification is required to current MPC layer. The case studies show that the proposed coordination scheme can substantially improve the performance of the existing decentralized control scheme, while it can take advantage of decentralized computing environment to ensure acceptable real-time calculation speeds.[1]

4.1 Background

Model predictive control (MPC) has gained extensive applications in industry for contrained control of multivariable processes. Bi-level (or two-stage) MPC technology has been widely applied in many industries. When MPC application is extended to large-scale operations, the deficiencies of the existing centralized and decentralized MPC schemes bring challenges to plant-wide MPC application. Plant-wide coordination of decentralized MPC has been identified as one of the most promising strategies to tackle the challenges ahead.

4.1.1 MPC Target Calculation

As is shown in Figure 4.1, the MPC framework can be divided into a steady-state calculation and a control calculation (or dynamic optimization), which are both executed at each control cycle (Qin and Badgewell, 2003; Kadam *et al.*, 2002). The target calculation determines the best achievable set-points, both for input and output variables; whereas, the trajectory along which the plant should be moved from one steady-state to the next is determined by the dynamic control calculation. In both calculations, a process model is required to perform the optimization. Note that in plant-wide control applications, the model and the problem size can be very large.

In industrial practice, a variety of optimization methods are applied to solve MPC target calculation problems, among which linear programming (LP) and quadratic programming (QP) are most commonly used (Qin and Badgewell, 2003). Many MPC technology products use a linear program to do the local steady-state optimization (e.g., the Connoisseur controller offered by Invensys, Inc.), while many use a quadratic program

---

[1] Parts of this chapter were presented/published in (Cheng *et al.*, 2004; Cheng *et al.*, 2005*b*).

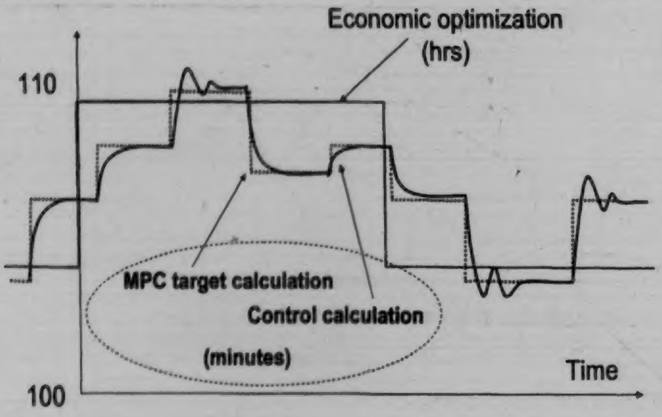


Figure 4.1: Bi-level MPC technology

to perform local steady-state optimization (e.g., the RMPCT, PFC, Aspen Target, MVC, process perfecter, etc.).

LP-based Target Calculation

For an individual MPC subsystem, which contains a steady-state target calculation and a dynamic control calculation, we can formulate an LP problem for the target calculation for time *k*:

$$\begin{aligned}&\min \quad z = \mathbf{c}^T\mathbf{x}(k)\\ &\text{subject to}\\ &\mathbf{A}_{eq}\mathbf{x}(k) = \mathbf{b}_{eq}(k)\\ &\mathbf{L}\mathbf{x}(k) \leq \mathbf{b}(k)\end{aligned} \tag{4.1}$$

where $\mathbf{x}(k) = [\mathbf{u}_s(k), \mathbf{y}_s(k)]$ is a vector of steady-state values (i.e., targets or setpoints) for the input and output variables for the subsystem. The equality constraints in (4.1) are taken from the linear dynamic model:

$$\mathbf{Y}(s) = \mathbf{G}(s)\mathbf{U}(s) + \mathbf{G}_d(s)\mathbf{D}(s) + \mathbf{E}(s) \tag{4.2}$$

which yields the the steady-state model:

$$\mathbf{y} = \mathbf{K}\mathbf{u} + \mathbf{K_d}\mathbf{d} + \mathbf{e} \tag{4.3}$$

where d are the disturbances and e is the unmeasured noise. The inequality constraints result from physical limitations on the inputs and outputs, such as actuator limits or production quality requirements. Other MPC target calculation formulations are possible, such as including model bias and soft output constraints (Kassmann *et al.*, 2000; Lestage *et al.*, 2002). These can easily be incorporated into the proposed method, but are omitted to simplify discussion.

QP-based Target Calculation

Assume that a set of optimal nominal "targets" $[\mathbf{y}^*, \mathbf{u}^*]$ has been given to each operating unit by an upper level optimizer, in order to follow the shifting optimum operating point and give appropriate corrections, an MPC target calculation for an individual operating unit can be formulated as the following constrained quadratic program (QP) (Ying and Joseph, 1999):

$$\begin{aligned}\min_{\mathbf{y}_{set},\mathbf{u}_{set}} \; z = {} & (\mathbf{y}_{set}(k) - \mathbf{y}^*)^T \mathbf{Q}_y (\mathbf{y}_{set}(k) - \mathbf{y}^*) + (\mathbf{u}_{set}(k) - \mathbf{u}^*)^T \mathbf{Q}_u (\mathbf{u}_{set}(k) - \mathbf{u}^*) \\ & + \mathbf{c}_y(\mathbf{y}_{set}(k) - \mathbf{y}^*) + \mathbf{c}_u(\mathbf{u}_{set}(k) - \mathbf{u}^*) + \boldsymbol{\epsilon}^T \mathbf{c}_\epsilon^T \mathbf{c}_\epsilon \boldsymbol{\epsilon}\end{aligned}$$

subject to:

$$\begin{aligned}\mathbf{y}_{set}(k) &= \mathbf{K}\mathbf{u}_{set}(k) + \mathbf{d}(k) \\ \mathbf{d}(k) &= \mathbf{d}(k-1) + \boldsymbol{\delta}(k) \\ \mathbf{y}_{min} - \boldsymbol{\epsilon} \leq \mathbf{y}_{set}(k) &\leq \mathbf{y}_{max} + \boldsymbol{\epsilon} \\ \mathbf{u}_{min} \leq \mathbf{u}_{set}(k) &\leq \mathbf{u}_{max} \\ \boldsymbol{\epsilon} &\geq 0\end{aligned} \tag{4.4}$$

where Q_y,Q_u, c_y, and c_u may be obtained from the upper level optimizer, and c_ϵ is a tuning parameter (Ying and Joseph, 1999); $\mathbf{y}_{set}(k)$ and $\mathbf{u}_{set}(k)$ are the setpoint values to be determined by the target calculation (i.e., they are the degrees of freedom for optimization); $\mathbf{d}(k)$ is the estimated disturbance and ϵ is a violation tolerance of the output constraints that ensures a feasible solution to the QP. In this work, we adopt a bias update strategy for d (i.e., $\boldsymbol{\delta}(k) = \mathbf{y}_m(k) - \mathbf{y}_{set}(k|k-1)$, where $\mathbf{y}_m(k)$ are the measured outputs at time k and $\mathbf{y}_{set}(k|k-1)$ is the prediction of outputs in the previous control execution), without loss of generality. The steady-state gain matrix **K** can be calculated via linearization of the nonlinear model used in an upper optimizing layer or abstracted from the linear model used by lower level MPC dynamic control. Note that the scope of problem (4.4) is a single operating unit.

4.1.2 Plant-wide MPC

With considerable development effort in recent years, there has been a trend to extend MPC to large-scale applications, such as plant-wide control. Two common paradigms for solving plant-wide MPC calculations are centralized and decentralized strategies. Centralized strategies may arise from the desire to operate the entire plant in an optimal fashion; whereas, decentralized MPC control structures can result from the incremental roll-out of automation systems. One major difference between these two extremes is the extent to which interactions among operating units are considered. An effective centralized or monolithic plant-wide MPC can be undesirable and difficult, if not impossible, to implement (Lu, 2003; Havlena and Lu, 2005). Such a scheme can exhibit poor fault-tolerance, can require a high performance centralized computational platform, and can be difficult to tune and maintain. Alternatively, in many chemical plants, large-scale control problems are solved by a group of MPC subsystems via decentralized schemes, in which each MPC controller takes care of a specified operating unit. As is shown in Figure 4.2, the decentralized MPC scheme yields the desired operability, flexibility and reliability, but may not provide an appropriate level of performance. In this work, reliability refers to the possibility that some control subsystems or portions thereof are able to function when other subsystems fail. Currently, decentralized MPC strategies are widely used due to their flexibility, reliability and ease of maintenance.

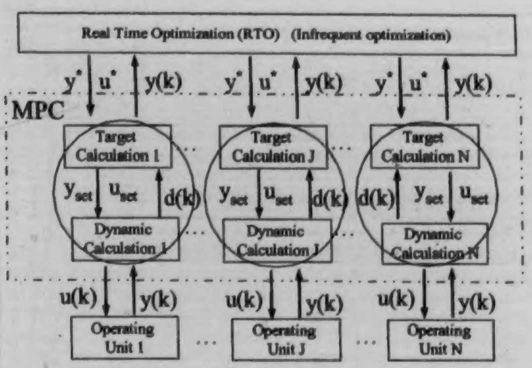

Real Time Optimization (RTO) (Infrequent optimization)
y*
u*
y(k)
MPC
Target Calculation 1
Target Calculation J
Target Calculation N
y_set
u_set
d(k)
Dynamic Calculation 1
Dynamic Calculation J
Dynamic Calculation N
u(k)
y(k)
Operating Unit 1
Operating Unit J
Operating Unit N

Figure 4.2: Decentralized MPC

4.1.3 Coordinated MPC

Although it has many good features as discussed, conventional decentralized schemes may not be able to provide the plant-wide optimum. In general, decentralized schemes approximate or ignore the interactions between operating units, while the downstream units treat upstream variations as external disturbances. Thus, the decentralized approach solves each subproblem in terms of its own objective function, which may not provide the plant-wide optimum operation. Lu (2003) claims that "the estimated latent global benefit for a typical refinery is 2-10 times more than what MPC by itself can capture". Thus, coordination of the unit-based controllers has been identified as having significant potential benefit (Lu, 2003; Havlena and Lu, 2005; Isaksson *et al.*, 2005), in other words, the key to exploiting the potential of decentralized control systems, yet still retaining its structure and advantages, is cooperation.

Usually, any limited cooperation between decentralized MPC controllers is through an upper level optimization, such as real-time optimization (RTO), at a sampling time of hours or days. Most RTO systems require waiting for the plant to be near a steady-state before they can execute. Disturbances or setpoint changes in the interval between two RTO executions may drive the optimum operations away from the targets given by the RTO system; thus, it may be necessary to perform re-optimization at a higher frequency than is possible through RTO to maintain optimum operations.

Recently, more effort has been spent on improving the performance of plant-wide decentralized control through coordination. Kumar and Daoutidis (2002) proposed a controller design framework using a time-scale decomposition approach, in which a supervisory controller deals with the slow-time-scale behavior and coordinates the distributed controllers, which deal with the fast-time-scale behavior. In Lu (2003), a cross-functional integration scheme was developed, in which a coordination "collar" performed a centralized target calculation for decentralized MPC. This idea matches the widespread belief among industrial practitioners (Scheiber, April 2004) that the trend toward decentralization will continue until the control system consists of seamlessly collaborating autonomous and intelligent nodes with a supervisory coordinator overseeing the whole process. In particular, the cooperation-based MPC scheme proposed by Venkat *et al.* (2004) is worthy of attention, which addressed *cooperation* between decentralized MPC controllers. Compared with the coordination framework in this work, both approaches aim to find plant-wide optimal operations rather than only to find a stabilizing solution. In cooperation-based MPC, where a state-space model is used, the objective function of each sub-controller involves the states and inputs of local unit and the impact from other units (i.e., the states and inputs of other units); similarly, the proposed coordination scheme involves an input-output model, and individual sub-controller has the objective function considering the inputs and outputs of the local unit and the impact from other units (i.e., the augmented variables which represent the interactions). Moreover, both control schemes include an iterative decision process within a single MPC execution, and the intermediate results are communicated. The decentralized controllers in Venkat *et al.* (2004) stand at equal status within their negotiation to achieve cooperation and the intermediate results are flowing among sub-controllers; however, this work addresses the cooperation between controllers through a well designed coordinator, which plays a specific role in not just transmitting information, but also in modifying the information that comes from the sub-controllers to ensure that the entire system finds the optimal operation. This work is focused on MPC target calculation, assuming that the dynamic control calculation has been appropriately formulated to ensure the required stability and robustness.

4.2 Coordination of Decentralized MPC

This section discusses a framework for the coordination of steady-state MPC target calculation level, which aims to provide a timely response to local or plant-wide disturbances and setpoint changes. Two key factors that determine the desirability of the coordinated MPC system are: computational efficiency of the coordination strategy to ensure a real-time solution; and required information flow load throughout the plant communication network. The proposed approaches exploit the existing plant computing, communication, and information systems with minimal modification, to provide significant performance improvement.

As has been discussed, the solution from the decentralized MPC target calculation may not be optimal with respect to the entire plant operation because of plant / model mismatch, which results from ignoring the interactions among operating units and other effects. Thus, some mechanism to take care of the interactions is desired. Note that, in this work, the term *unit-based MPC* refers to the decentralized MPC subsystems developed for individual operating units.

4.2.1 Coordination through a Coordinator

Figure 4.3 shows a coordination-based MPC system, where the coordinator is designed to be responsible for ensuring that the effects of interactions are incorporated into the overall control strategy. The task of the coordinator is to ensure that the coordinated system finds the optimal plant operations. A coordinator can be designed by considering different kinds of interactions among operating units. Such interactions can be formulated as linking constraints.

4.2.2 Identification of Linking Constraints

Recall that the two key factors required of an efficient coordination are the computational efficiency of the coordination strategy and the required information flow throughout the plant communication network. In the previous section we dealt with the coordination, and in this section we are going to introduce two message construction approaches based on

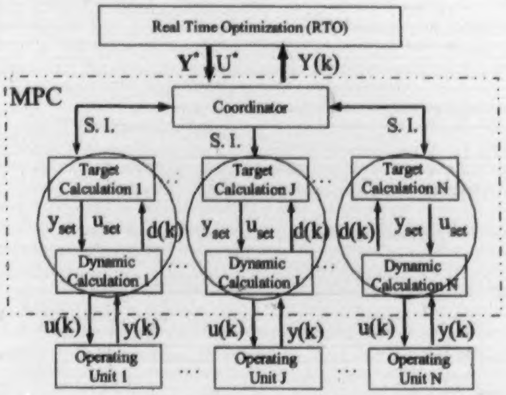

Real Time Optimization (RTO)
Y*
U*
Y(k)
MPC
Coordinator
S. I.
S. I.
S. I.
Target Calculation 1
Target Calculation J
Target Calculation N
y_set
u_set
d(k)
Dynamic Calculation 1
Dynamic Calculation J
Dynamic Calculation N
u(k)
y(k)
Operating Unit 1
Operating Unit J
Operating Unit N

Figure 4.3: Coordinated, decentralized MPC

different interaction modeling methods, which ensure reasonable data traffic through the plant communication network. In this section, two methods to establish linking constraints for interactions are discussed, such as the interstream consistency (Cheng *et al.*, 2004) and off-diagonal element abstraction (Cheng *et al.*, 2005*a*).

Interstream Consistency

In many cases, the interactions between two operating units can be modeled by equating the appropriate output variables from the upstream unit and the input variables to the downstream unit. Shown in Figure 4.4, the hexagon labeled "A" represents the process interstream connecting individual operating units.

In formulating the subproblems, those streams connecting different operating units are torn and consistency relationships can be used to model the interactions between different units. Recall a block-wise linear programming problem that has been converted to Simplex standard form:

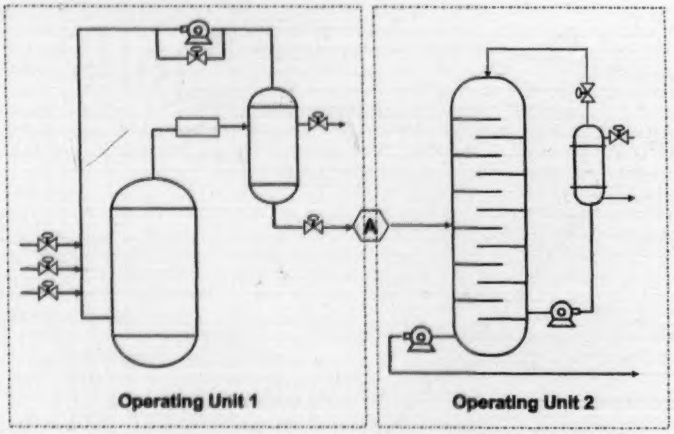


Figure 4.4: Demonstration of interstream consistency

$$\min \quad z_1 = \sum_{i=1}^{p} \mathbf{c}_i^T \mathbf{x}_i$$

subject to

$$\sum_{i=1}^{p} \mathbf{A}_i \mathbf{x}_i = \mathbf{b}_0 \tag{4.5}$$

$$\mathbf{B}_i \mathbf{x}_i = \mathbf{b}_i \tag{4.6}$$

$$\mathbf{x}_i \geq \mathbf{0} \quad i = 1, 2, ..., p$$

where (4.5) represents the linking constraints associated with p subproblems, and the constraints in (4.6) are the local constraints of independent subproblems. In this case, assume that we have p separate operating units, each of which is controlled by one MPC subsystem. By introducing interprocess stream consistency as the linking constraints (4.5), we can formulate an LP problem that includes constraints (4.5) and (4.6) by incorporating those decentralized target calculation problems.

Off-diagonal Element Abstraction

As previously discussed, this work assumes the controlled variables (CVs) and manipulated variables (MVs) have been specified and grouped in a unit-based sense (i.e., we are coordinating an existing set of unit-based MPC systems). Then the gain matrix for the entire plant with N operating units is:

$$\mathbb{A} = \begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} & \dots & \mathbf{K}_{1N} \\ \mathbf{K}_{21} & \mathbf{K}_{22} & \dots & \mathbf{K}_{2N} \\ \dots & \dots & \dots & \dots \\ \mathbf{K}_{N1} & \mathbf{K}_{N2} & \dots & \mathbf{K}_{NN} \end{bmatrix} \tag{4.7}$$

The matrix A can be ordered such that a unit-based implementation of MPC as in problem (4.4) uses the block-diagonal information $\mathbf{K}_{ii}$ of the plant model in its calculations. In such a case, the off-diagonal blocks may be treated as disturbances. This way of dealing with the off-diagonal information can result in undesirable closed-loop behavior, when the interactions are significant. Note that the plant-wide model is:

$$Y(k) = \mathbb{A}U(k) + D(k) \tag{4.8}$$

where $Y(k)$ and $U(k)$ are the CVs and MVs of N local operating units concatenated into vectors; $D(k)$ is the concatenation of local disturbance variables (DVs). This is equivalent to:

$$\mathbf{y}_i(k) = \mathbf{K}_{ii}\mathbf{u}_i(k) + \mathbf{e}_i(k) + \mathbf{d}_i(k) \tag{4.9}$$

$$\mathbf{e}_i(k) - \sum_{j=1}^{N} \mathbf{K}_{ij}\mathbf{u}_j(k) = \mathbf{0} \quad j \neq i \tag{4.10}$$

The auxiliary variable $\mathbf{e}_i$, which is an abstraction of the off-diagonal elements, represents the influence of the inputs of other operating units on the local system. In the proposed coordination scheme, a coordinator will be developed to handle the constraints (4.10) and drive the auxiliary variable $\mathbf{e}_i$ to the values corresponding to the plant-wide optimum operations. In this case, in each unit-based MPC target calculation, the auxiliary vector $\mathbf{e}_i$ is treated as a decision variable, since equations (4.10) are not included in each unit-based MPC calculation.

4.3 Dantzig-Wolfe Decomposition and Plant-wide MPC

For industrial plant-wide control, this work provides a new formulation of plant-wide MPC target calculation to achieve plant-wide optimum operations. A novel application of the Dantzig-Wolfe decomposition has been proposed in the design of coordination system for solving the plant-wide MPC problem.

4.3.1 Illustrative Case Studies

In this section, we illustrate the implementation of the proposed coordination scheme through two case studies: the first is used to investigate the interstream consistency approach and the second investigates the off-diagonal element abstraction approach.

Case Study 1

Let us consider a system shown in Figure 4.5. The normalized gains for the system are given in (4.11) through (4.13). An identity matrix is chosen for $\mathbf{K}_d$ in (4.3) assuming that the disturbances influence the outputs directly. The locations where the disturbances entering the plant are shown as dashed lines in Figure 4.5.

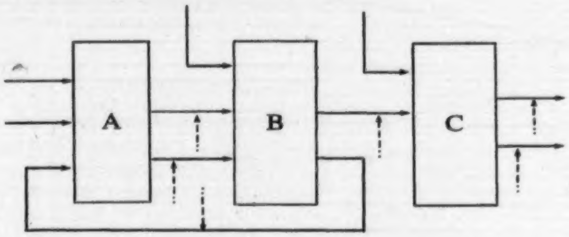


Figure 4.5: Interacting MIMO operating units

$$K_A = G_A(0) = \begin{bmatrix} 0.4 & 0.6 & 0.1 \\ 0.5 & 0.4 & 0.1 \end{bmatrix} \tag{4.11}$$

$$K_B = G_B(0) = \begin{bmatrix} 0.3 & 0.4 & 0.3 \\ 0.1 & 0.2 & 0.1 \end{bmatrix} \tag{4.12}$$

$$K_C = G_C(0) = \begin{bmatrix} 0.7 & 0.3 \\ 0.6 & 0.5 \end{bmatrix} \tag{4.13}$$

Each operating unit has its own objective, which is a subset of information used by plant-wide optimizers, and the profit function cost coefficients are:

$$\mathbf{c}_A^T = \begin{bmatrix} -1 & -1 & -1 & 3 & 3 \end{bmatrix} \tag{4.14}$$

$$\mathbf{c}_B^T = \begin{bmatrix} -1 & -1 & -1 & 3 & 3 \end{bmatrix} \tag{4.15}$$

$$\mathbf{c}_C^T = \begin{bmatrix} -1 & -2 & 5 & 5 \end{bmatrix} \tag{4.16}$$

So for each operating unit, by tearing the interprocess stream, a linear program for the k^{th} target calculation is:

$$\min \quad \mathbf{c}_j^T \mathbf{x}_j(k)$$

subject to

$$\mathbf{K}_j \mathbf{x}_j(k) = \mathbf{b}_{eq}^j(k) \tag{4.17}$$

$$\mathbf{L}_j \mathbf{x}_j(k) \leq \mathbf{b}^j(k), \quad j = A, B, C \tag{4.18}$$

where L_j stands for the coefficient matrix associated with all the inequality constraints when it is in standard form. The R.H.S. of the equality constraints $\mathbf{b}_{eq}^j(k)$ represent the updated model bias at each target calculation execution. The R.H.S. of the inequality constraints $\mathbf{b}^j(k)$ contain the lower bounds (lb) and upper bounds (ub) of the variables in the operating units. The bounds on the variables in this case study are shown in equation (4.19) and (4.20).

$$\mathbf{lb} = [\ 0.3\ \ 0.3\ \ 0.15\ \ 0.45\ \ 0.4\ \ 0.45\ \ 0.4\ \ 0.3\ \ 0.45\ \ 0.15\ \ 0.45\ \ 0.3\ \ 0.45\ \ 0.5\ ] \tag{4.19}$$

$$\mathbf{ub} = [\ 0.5\ \ 0.5\ \ 0.25\ \ 0.55\ \ 0.5\ \ 0.55\ \ 0.5\ \ 0.5\ \ 0.55\ \ 0.25\ \ 0.55\ \ 0.5\ \ 0.55\ \ 0.6\ ] \tag{4.20}$$

Three different MPC strategies, centralized MPC, decentralized MPC and coordinated, decentralized MPC, are implemented to evaluate their abilities to track the changing optimum in steady-state target-calculations. For the centralized MPC target calculation, an LP problem is formulated treating all the inputs and outputs, including interprocess interactions, as decision variables. For the decentralized MPC scheme, separate LP problems are formulated by passing the upstream decisions to downstream units as disturbances. Finally, the coordinated MPC target calculation incorporates the linking constraints in modeling the interactions and solves the RMP and independent subproblems iteratively.

In our case study, unknown disturbances are generated by filtering random series of uniformly distributed variates in order to restrict these disturbances within the interval ± 0.05. These unknown disturbances are directly imposed on the outputs when the optimized targets are implemented in our simulation.

$$\mathbf{d}(t) = \frac{1}{1 + c_1 q^{-1} + c_2 q^{-2}} \mathbf{e}(t) \tag{4.21}$$

By using the autoregressive models in equation (4.21) as simplified disturbance models, we predict one-step ahead disturbances based on past information. The estimated disturbances used to update the disturbance model in (4.21) are calculated by comparing the measured outputs and model predictions at every control execution. At the current control calculation, the parameters, c_1 and c_2, in the disturbance model are estimated using the estimated disturbances in the past 10 control execution periods. The one-step ahead disturbances are predicted using the estimated c_1 and c_2. Then the process models are updated using the predicted disturbances. The steady-state targets are then calculated by using the updated process models.

The following accumulated profit function is defined for performance comparison:

$$P = \sum Z(k) * T_s - \sum_{k=i} V(k) * T_s \tag{4.22}$$

where $Z(k)$ is the actual profit per unit time from the k^{th} target calculation; T_s is the sampling period between two target calculations; and $V(k)$ represents the penalty for constraint violations when we implement the calculated targets:

$$V(i) = \mathbf{w}^T \mathbf{y}_v(i) \tag{4.23}$$

where **w** is a specified penalty vector, which is used in all three cases; and the violation of output constraints $\mathbf{y}_v(i)$ is defined as:

$$\mathbf{y}_v(i) = \begin{cases} \mathbf{y}_{act}(i) - \mathbf{y}_{max}, & \text{if } \mathbf{y}_{act}(i) \geq \mathbf{y}_{max} \\ \mathbf{y}_{act}(i) - \mathbf{y}_{min}, & \text{if } \mathbf{y}_{act}(i) \leq \mathbf{y}_{min} \end{cases} \tag{4.24}$$

where $\mathbf{y}_{act}(i)$ is the actual output vector when the calculated targets are implemented in the process; while $\mathbf{y}_{max}$ and $\mathbf{y}_{min}$ are subsets of the upper and lower bounds given in equations (4.19) and (4.20).

A benchmark is defined for comparing the performance of different MPC target calculation strategies. The benchmark used for comparison is defined as the maximum profit achieved when the plant is operated at the true optimum, which is calculated using the perfect process model and exact knowledge of disturbances. Although this maximum profit is not achievable, it is a useful basis for performance comparison.

Figure 4.6 shows the profit achieved as a function of control execution using different MPC steady-state target calculation strategies. The coordinated target calculation gives the same achievable optimum as the centralized MPC scheme does[2], while the decentralized scheme yields a suboptimal operation as interactions are ignored in the calculation.

Table 4.1 compares the performance of different MPC strategies for a simulation of 150 target calculation executions. From table 4.1, we can see that the centralized and

---

[2] When all the interactions considered in centralized scheme are handled by the coordinator in the coordinated scheme, the two approaches will provide the same solution; however, if any interactions are ignored in the coordinated scheme, the solution from the coordinated scheme may deviate from the centralized scheme.

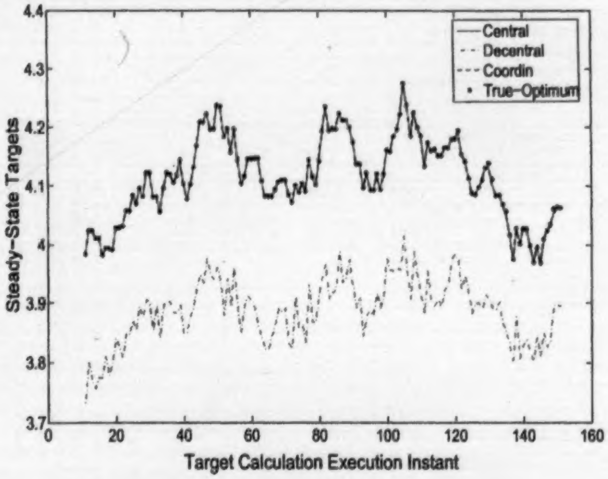


Figure 4.6: Calculated targets by different approaches

the coordinated, decentralized target calculation give the same best achievable profit and achievable ratio to the true optimum, while the fully decentralized target calculation only captures approximately 94.48% of the maximum profit.

Table 4.1: Performance comparison of different plant-wide MPC target calculation approaches for interacting MIMO systems

	Centralized	Decentralized	Coordinated	True. Opt.
Profit	5800.9	5481.7	5800.9	5801.7
Achiev. Ratio	99.98	94.48	99.98	100
Prob. Dimension	46×42	$15 \times 15 \times 3$	$7 \times 7 + 15 \times 15 \times 3$	NA

* All the simulations were performed in MATLAB® 6.5 on a Pentium III 1.0G Hz and 512M RAM machine.

Table 4.1 also reports the problem sizes for different steady-state target calculation strategies. The problem size is defined as the size of the coefficient matrix in the LP standard form used in the Simplex method. Therefore, slack and excess variables are added to the convert the inequality constraints to equality constraints, and the columns of the coefficient matrices are augmented to incorporate the slack and excess variables. We can see that the centralized scheme has the largest problem size, which will grow significantly when the dimension of separate problems and the number of operation units in the flowsheet increase. The problem size for the decentralized MPC is reported as the dimension of the largest subproblem multiplied by the number of units. For the coordinated scheme, the problem size is expressed as the addition of two components, the dimension of RMP (the coordinator) and the problem dimension in the decentralized scheme (the coordinated parts).

This case study shows that interstream consistency can be used for interaction modeling, and when such interactions are handled by the coordinator, the resulting coordinated, decentralized control system does produce significant improvement on the plant-wide performance. As such, it provides an approach to plant-wide control that does not require a centralized computing environment.

Case Study 2

This case study is to illustrate the application of the off-diagonal element abstraction method for coordinating decentralized MPC by using the Dantzig-Wolfe decomposition principle (Cheng *et al.*, 2005*a*).

Shown in Figure 4.7, a generic process network is used for this case study. Described as follows, the overall process network can be represented by an 8-input and 6-output model G, which is a linearization of the process around an operating point $[\mathbf{y}_0, \mathbf{u}_0]$:

$$\begin{aligned} \mathbf{y}_0 &= [5, 3, 4, 2, 8, 10] \\ \mathbf{u}_0 &= [0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5] \end{aligned} \tag{4.25}$$

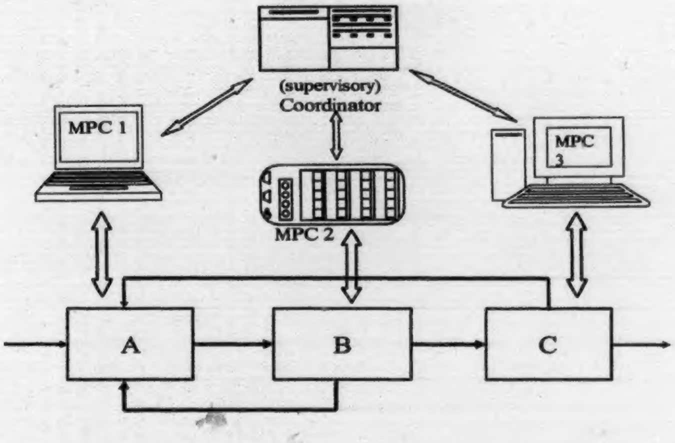


Figure 4.7: An interacting MIMO unit network

$$\mathbf{G}_0^T = \begin{pmatrix} -0.88 & 1.49 & 0 & -2.36 & 0 & -1.4 \\ 1.13 & -0.5 & 0 & 0.24 & 0 & -0.26 \\ 1.49 & 2.59 & 0 & -1.19 & 0 & 0.77 \\ 0 & 0.55 & -0.42 & -0.32 & 0 & 1.48 \\ 0 & 3.03 & 0.4 & -0.97 & 0 & 1.12 \\ 0 & 2.56 & 0 & 0 & -0.25 & 0.06 \\ 0 & 0.66 & 0 & 0 & -2.1 & -0.55 \\ 0 & 0.29 & 0 & 0 & -0.28 & -0.61 \end{pmatrix}$$

where $\mathbf{G}_0$ is the steady-state gain matrix of the process model $\mathbf{G}$. The flowsheet was originally decomposed into three operating units, each of which has two output variables. Further, unit A and unit C have three manipulated variables, while unit B has two. Each operating unit has its own objective, which is a subset of information used by plant-wide optimizers. In this maximization problem, the profit function cost coefficients are:

$$\mathbf{c}_A^T = \begin{bmatrix} 2 & 3 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{c}_B^T = \begin{bmatrix} 1 & 3 & 0 & 0 \end{bmatrix} \quad \mathbf{c}_C^T = \begin{bmatrix} 4 & 7 & 0 & 0 & 0 \end{bmatrix}$$

where the objective functions are only related to output variables.

The decentralized MPC controllers use incomplete process information and ignore the interactions. Using the *off-diagonal element abstraction* method, we can employ an auxiliary term to represent the interactions and augment them in the model of each decentralized MPC system as discussed in (4.9). Note that, since the off-diagonal elements are relatively sparse, the dimension of $\mathbf{e}_m$ is only three. In this case study, we set the upper and lower bounds of the decision variables within ±10% interval of the nominal operating point, while treating $\mathbf{e}_m$ as unrestricted variables.

Within this configuration, we can specify an information package which flows on the communication network that connects the coordinator and individual control systems. Similar to Figure 4.3, at every coordination round, each decentralized MPC system submits to the coordinator a unit-wide optimal solution $\mathbf{x}_i = [\mathbf{y}_i, \mathbf{u}_i, \mathbf{e}_m^i]$ and corresponding objective function value f_i. As soon as it receives all the proposals, the coordinator executes a linear program to solve the master problem. Then the coordinator records the solution λ_{ij} and sends sensitivity information $[\boldsymbol{\pi}, \gamma_i]$ to the decentralized MPC. Here, $\boldsymbol{\pi}$ is related to the auxiliary variable $\mathbf{e}_m^i$, which can reflect the gap between the plant-wide optimal solution and unit-wide solution. In other words, when an optimal $\mathbf{e}_m^i$ is obtained through coordination, the plant-wide optimum is reached.

To simplify the discussion in our case study, we assume accurate modeling and noise-free simulation. Therefore, in terms of plant-wide optimum, a centralized controller provides benchmark performance. With the above information, one execution of plant-wide target calculation is performed with three optimization schemes, the *centralized, decentralized, and coordinated* schemes, respectively.

Table 4.2: Performance comparison of different plant-wide MPC target calculation approaches for interacting MIMO unit network

	Centralized	Decentralized	Coordinated
Profit Function Value	134.674	130.035	134.674
Achievability Ratio	100	96.56	100

* The simulations were performed in MATLAB® 7.0 on a Pentium III 1.0G Hz, 512M RAM machine.

Table 4.2 provides simulation results for the comparison of the performance of different control strategies. We can see that the centralized and the coordinated target calculation schemes give the same achievable profit as the benchmark optimum, while the fully decentralized steady-state target calculation only captures around 96.56% of the maximum profit.

A key point drawn from this study is that the proposed approach may require far less capital investment to gain equal performance increases, in comparison to implementation of a new centralized, plant-wide MPC. Again, it also provides an approach to plant-wide control that can take advantage of the existing distributed computing environment.

4.4 Price-driven Coordination and Plant-wide MPC

For the coordination of QP-based MPC target calculation on a pulp mill process. In this section, a case study on the plant-wide control of a pulp mill process is performed to illustrate the effectiveness of the proposed price-driven coordination approach.

4.4.1 A Pulp Mill Benchmark Process

The pulp mill benchmark problem given in Castro and Doyle III (2004*a*) was recently published and proposed for investigating the efficacy of control and optimization approaches. As is shown in Figure 4.8, the pulp mill model includes the fiber-line and the chemical recovery loop. The primary goal of the pulp mill is to produce wood pulp of a given Kappa number or brightness while minimizing energy costs, utilities and chemical make-up streams.

The benchmark is based on a nonlinear dynamic mathematical model with approximately 8200 states and a total of 142 inputs (82 MVs and 60 DVs) and 114 outputs (40 in fiberline and 74 in chemical recovery). This model was developed to be approximately 200 times faster than the real process. The control objectives, modes of operation, process constraints and measurements are all defined in Castro and Doyle III (2004*a*).

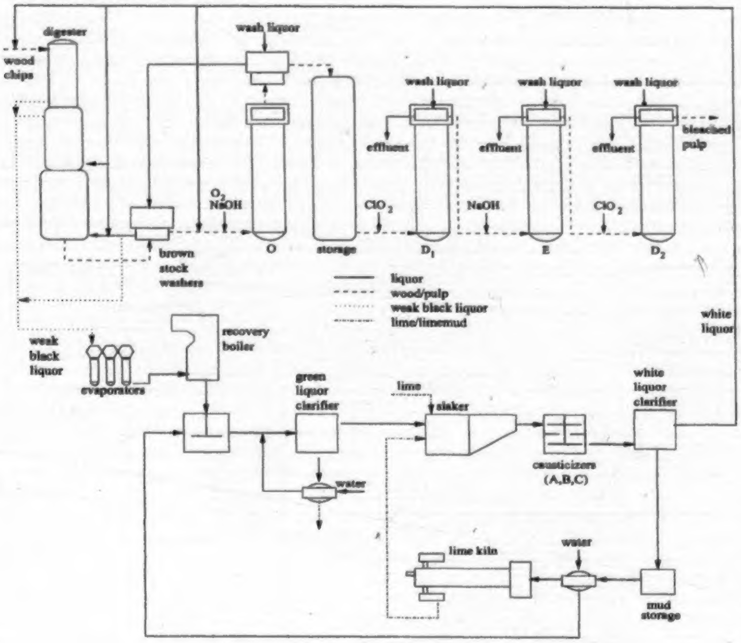

digester
wood chips
wash liquor
wash liquor
wash liquor
wash liquor
effluent
effluent
effluent
bleached pulp
O_2 NaOH
ClO_2
NaOH
ClO_2
brown stock washers
O
storage
D_1
E
D_2
liquor
wood/pulp
weak black liquor
lime/limemud
white liquor
weak black liquor
evaporators
recovery boiler
green liquor clarifier
lime
slaker
causticizers (A,B,C)
white liquor clarifier
water
lime kiln
water
mud storage

Figure 4.8: Pulp mill benchmark process

4.4.2 Price-driven Coordination for Plant-wide Control

In Castro and Doyle III (2004*b*), a decentralized control system has been proposed. At the unit level, it involves two control layers: unit-based MPC and decentralized regulatory control loops. This case study focuses on the MPC layer.

The existing MPC consists of four separate controllers, one each for the digester and oxygen reactor, the bleach plant, the evaporators, and the lime kiln/recaust areas, respectively. In their configuration, the MPC layer only contains the dynamic control calculation stage and involves totally 21 CVs and 20 MVs. The MPC is designed to track the set-point trajectories given by an upper level optimization.

This study uses the decentralized, two-stage MPC system discussed in Ying and Joseph (1999) and takes the formulation given in problem (4.4) for target calculation. The MPC control calculations in this paper are as given by Castro and Doyle III (2004*b*). In the unit-based MPC target calculation, the interactions between units were ignored. Thus, the gain matrix $\mathbf{K}$ in (4.4) is actually $\mathbf{K}_{ii}$, the block-diagonal elements of the overall-plant gain matrix $\mathbb{A}$. The effect of off-diagonal elements was treated as disturbances, through $\mathbf{d}(k)$. The bounds for variables are the same as in the dynamic control calculation, and the weightings $\mathbf{Q}_y$ and $\mathbf{c}_y$ are given in Table 4.3.

Plant-wide Coordination

Since the focus is on the MPC target calculation, the plant-wide linear steady-state model matrix $\mathbb{A}$ in (4.7), from the MVs to CVs, is obtained via step response tests to ensure that the steady-state gains are consistent with the dynamic simulation. In this work, bias update strategy discussed in Ying and Joseph (1999) was used to compensate for disturbance and model mismatch, *etc.*

Using the price-driven coordination strategy given in Chapter 3, the MPC target calculation for a local operating unit can be modified as:

$$\begin{aligned}\min_{\mathbf{y}_{set},\mathbf{u}_{set}} \quad z_3 = {} & (\mathbf{y}_{set}(k)-\mathbf{y}^*)^T\mathbf{Q}_y(\mathbf{y}_{set}(k)-\mathbf{y}^*)+(\mathbf{u}_{set}(k)-\mathbf{u}^*)^T\mathbf{Q}_u(\mathbf{u}_{set}(k)-\mathbf{u}^*) \\ & +\mathbf{c}_y(\mathbf{y}_{set}(k)-\mathbf{y}^*)+\mathbf{c}_u(\mathbf{u}_{set}(k)-\mathbf{u}^*)+\boldsymbol{\epsilon}^T\mathbf{c}_\epsilon^T\mathbf{c}_\epsilon\boldsymbol{\epsilon}-\mathbf{p}^T\mathbf{e}(k)\end{aligned}$$

subject to:

Table 4.3: Important CV weightings

Controlled variables	$\mathbf{Q}_y/100$	$\mathbf{c}_y$
production rate	1.5	-80
digester kappa No.	1.5	0
oxygen reactor kappa No.	1.0	0
oxygen reactor caustic flow	1.0	0
oxygen reactor steam flow	0.5	0
oxygen reactor coolant flow	0.75	30
E kappa No.	1.0	0
D_2 brightness	1.0	0
slaker temperature	1.0	0
kiln O_2 excess %	1.0	0
kiln fuel flow	0.5	30

$$\begin{aligned} \mathbf{y}_{set}(k) - \mathbf{K}\mathbf{u}_{set}(k) &= \mathbf{e}(k) + \mathbf{d}(k) \\ \mathbf{d}(k) &= \mathbf{d}(k-1) + \boldsymbol{\delta}(k) \\ \mathbf{y}_{min} - \boldsymbol{\epsilon} \leq \mathbf{y}_{set}(k) &\leq \mathbf{y}_{max} + \boldsymbol{\epsilon} \\ \mathbf{u}_{min} \leq \mathbf{u}_{set}(k) &\leq \mathbf{u}_{max} \\ \boldsymbol{\epsilon} &\geq 0 \end{aligned} \tag{4.26}$$

where the subscript i is omitted for simplicity. Note that a price vector $\mathbf{p}$ is introduced into the objective function and an auxiliary term $\mathbf{e}(k)$ is included in the unit model. To find the equilibrium price vector $\mathbf{p}^*$, the proposed price-adjustment scheme requires the solution of:

$$\boldsymbol{\Delta}(\mathbf{p}) = \mathbf{e}_i - \sum_{j=1}^{N} \mathbf{K}_{ij}\mathbf{u}_j \quad i = 1...N \;\; j \neq i \tag{4.27}$$

The equilibrium price vector $\mathbf{p}^*$, which satisfies $\boldsymbol{\Delta}(\mathbf{p}) = 0$, is the solution to problem (4.26). This procedure usually takes a few communication cycles between the coordinator and subsystems. When the price vector is appropriately updated, the unit-based MPC solutions will converge to the plant-wide optimum.

Closed-loop Performance

This section compares three control schemes: the centralized, the decentralized, and the coordinated, decentralized MPC target calculation. The centralized optimization scheme uses the entire plant-wide gain matrix and is used to define the performance benchmark for our study.

The control objectives are to closely track the setpoints given by an upper level optimization, while maximizing production rate and minimizing oxygen reactor coolant flow and kiln fuel flow. In this case study, the overall objective function for the plant is defined as a linear combination of those objectives with weightings given in Table 4.3. The optimization problems in all of the schemes are formulated as minimization problems. The weightings for the MVs used in all MPC control schemes are adopted from Castro and Doyle III (2004*b*).

Results are provided for 8000 minutes of closed-loop simulation. Please note that the abscissa in Figure 4.9 and 4.10 has the units hours. The disturbance set imposed on the process was adopted from Castro and Doyle III (2004*b*). The coordinated scheme provides identical performance to that of the centralized scheme. Thus, only the closed-loop responses for the coordinated scheme and the original decentralized scheme are shown in Figure 4.9 and 4.10. In this study, the decentralized scheme exhibits significant offset from the optimum plant operations.

Table 4.4 reports the profit/cost function values and computational times for all three control schemes.

Table 4.4: QP-based MPC target calculation performance comparison

Control Schemes	Value Function	Optimization Time* per Control Interval
Centralized	1.22×10^5	0.06 s
Unit-based	1.32×10^5	0.04 s
Coordinated	1.22×10^5	0.14 s

* The simulations were performed in MATLAB® 6.1 on an Athlon 1.4G Hz, 1024M RAM machine.

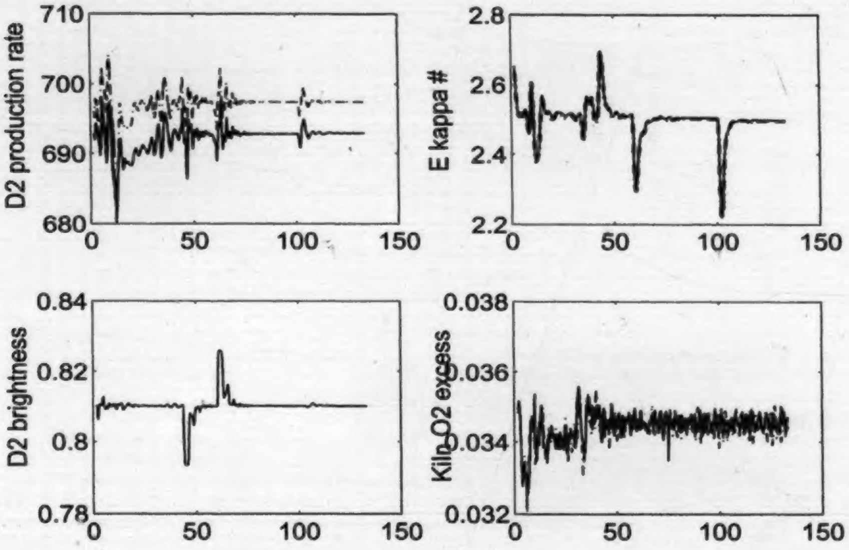


Figure 4.9: Closed-loop response 1: solid line (coordinated); dash line (decentralized)

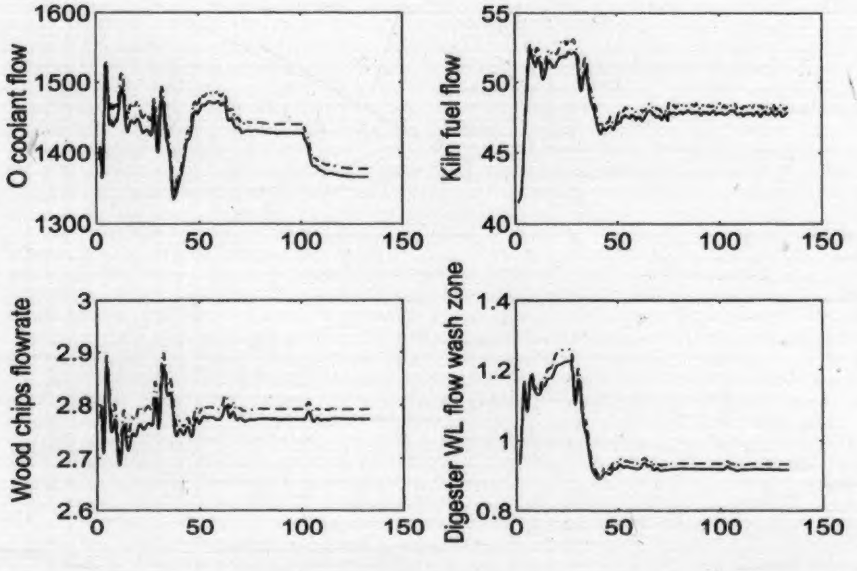


Figure 4.10: Closed-loop response 2: solid line (coordinated); dash line (decentralized)

Note that the accumulated value function is a time-integration of the objective function evaluated using the measured process variables. The optimization time is an average value based on the observed computational times. As we have defined the value function of the centralized scheme as a benchmark, we can see that the coordinated decentralized MPC provides the same plant-wide operations, which produces an 8.2% improvement on that of the decentralized scheme.

The computational effort is also reported in table 4.4 as the average optimization time for each MPC execution interval. Since all the simulations were performed in a single-processor machine, we approximate the computational load in the decentralized scheme by averaging the sum of maximum times spent on solving each individual subproblem at one interval. The computational load of the coordinated scheme consists of the time taken for coordination and the time used for solving the most computationally intensive subproblem at each iteration. It should be noted that the overall time for the coordinated case is nearly a factor of 2 more than the other two cases. This was accomplished without using any of the available coding techniques for decentralized computing. Thus, the above computational efficiency suggests some promise of the proposed coordination strategy for industrial on-line application.

In the case study, the optimization problems involve dozens of decision variables and hundreds of constraints. The coordinated MPC scheme provides solutions at a reasonable computational speed and as a result, exhibits a good trade-off between accuracy, reliability and computational load.

Remarks: Some Implementation Issues

In the case study, the sampling time for target calculation is chosen as 10 minutes, which is the least common multiple of the sampling times of MPC subsystems. Based on our observations from simulations, the selection of sampling time for coordination should also depend on the frequencies of the disturbances that the control system must deal with.

In general, good initial points can substantially enhance the efficiency of optimization. In this study, it was found that the equilibrium price vector from the previous execution worked very well as an initial guess for the current target calculation.

4.5 Chapter Summary

In many plant-wide control and optimization applications, a large-scale process model is decomposed into several smaller subsystems and a controller is developed for each subsystem. This may lead to a decentralized unit-based MPC framework, which may not be able to provide the plant-wide optimum operations because of its failure to consider the interactions between subsystems in decentralized MPC calculations. The coordination of the unit-based MPC systems has been identified as having significant potential benefit.

In this chapter, a new approach to coordinating decentralized MPC target calculation is proposed by taking advantage of the Dantzig-Wolfe decomposition algorithms. Several message construction methods are proposed for coordination system design, in which the constraints associated with multiple units can be incorporated. We have developed a framework of designing a coordination system for decentralized MPC with minor modification to current MPC layer. Our work shows that the proposed coordinated target calculation scheme substantially improves the performance of the existing decentralized control scheme, while it can utilize decentralized computing environment to ensure acceptable real-time calculation speeds.

In this chapter, with the proposed price-driven coordination method and off-diagonal element abstraction technique, a coordinator is developed to handle interactions between operating units, based on existing unit-based MPC systems. This results in a coordinated, decentralized MPC framework. In the study of pulp mill problem, the proposed coordinated, decentralized MPC shows a significant improvement in the plant operations in comparison with the existing decentralized MPC. The enhanced price-driven coordination algorithm shows promise in providing an acceptable online calculation speed for solving industrial plant-wide MPC control and optimization problems, which implies that the proposed coordinated, decentralized MPC framework may be a viable technology for plant-wide MPC applications.

"If we knew what we were doing it wouldn't be research. "

– by Albert Einstein

5

Extension to Large-scale Mixed-integer Programming

Optimization of many operations in industry takes the form of large-scale mixed-integer programming (MIP)[1], so it is of value to extend our research to mixed-integer optimization problems. In this chapter, our study is extended to decomposition and coordination strategies for solving large-scale MIP problems, particularly mixed-integer linear programming (MILP) problems. Subgradient optimization techniques are widely used in solving large-scale MILP problems[2] because of their ease of implementation; however, their convergence behavior needs improvement for the applications in which high speed computation is required. In this work, in order to enhance the computational efficiency of the subgradient optimization methods, new heuristics are proposed to adjust the search direction. A complexity study is performed and provides insight into the scaling behavior of the proposed subgradient optimization algorithm. In addition, to make use of the solution results from the subgradient optimization methods, primal solution recovery techniques are investigated. An interior path searching method is proposed for primal solution recovery by using the Dantzig-Wolfe decomposition strategy. With the improved subgradient optimization algorithm and the primal solution recovery heuristic, a decentralized optimization framework is developed and applied to a distributed decision support system for truck allocation in mining operations. The case study shows the proposed optimization framework may be a viable technique for solving industrial MILP problems.

---

[1] In this work, mixed-integer programming generally includes mixed-integer linear programming (MILP), mixed-integer quadratic programming (MIQP), mixed-integer nonlinear programming (MINLP), and binary integer programming (BIP).

5.1 Background

Many optimization problems involve a combination of continuous and discrete variables, and a partial list includes process synthesis (e.g., heat exchanger networks, etc.), design, scheduling, and planning of batch processes, interaction of design and control, and hybrid control systems (Floudas, 1995; Bemporad and Morari, 1999; Stursberg and Panek, 2002). Moreover, the Assignment problem, Knapsack problem, Set Covering problem, and Vehicle Routing problem discussed in Beasley (1993) are typical mixed-integer (or binary integer) programming problems abstracted from many practical problems in various areas. Impressive collections of MIP applications, such as scheduling, supply chain and hybrid control applications, can be found in Biegler and Grossmann (2004), Grossmann and Biegler (2004).

In the following subsections, we are going to briefly discuss the complexity issues in mixed-integer programming and some known MIP decomposition strategies, which take advantage of the special structure of a large-scale MIP problem.

---

[2] Subgradient optimization techniques are not restricted to solving MILP problems, and have been used in MIQP and MINLP problems.

5.1.1 Complexity Issues in MIP

The solution of large-scale MIP problems is very challenging, because they combine all of the difficulties of their subclasses. For instance, large-scale MILP and MIQP problems present the combinatorial nature of integer programs and the difficulty in solving large-scale LP and QP problems. Any choice of integer combination (e.g., binary 0 or 1 as a common case) from the elements of the discrete feasible region results in an LP or QP problem with continuous variables. If one follows a full enumeration approach to solving an MIP problem, for example to solve a "0-1" MILP problem, it grows exponentially in time with respect to its computational effort (Floudas, 1995). For instance, one hundred binary variables would result in 2^{100} possible combinations.

Wolsey (1998) provides a summary of the complexity analysis results for several classes of mixed-integer programming problems. Some of them fall into the category of NP-complete problems, and some of them are NP-hard. Here, "NP" means verifiable in nondeterministic polynomial time, and "NP-hard" means at least as hard as any NP-problem, although it might, in fact, be harder. An "NP-complete" problem is both NP (verifiable in nondeterministic polynomial time) and NP-hard (any other NP-problem can be translated into this problem).

5.1.2 Decomposition Strategies for Mixed-integer Programming

In previous chapters, several methods for solving continuous optimization problems have been investigated, including the Dantzig-Wolfe decomposition principle (Lasdon, 2002; Cheng *et al.*, 2004) for linear programming (LP), the price-driven (or auction-based) coordination method (Jose and Ungar, 1998*b*; Cheng *et al.*, 2005*b*) for quadratic programming (QP). The Benders' decomposition method (Benders, 1962; Lasdon, 2002) was originally developed for solving MIP problems. In its most common implementation, it decomposes the decision variable space into continuous and discrete parts, which are handled by a master problem and a subproblem, respectively. The master problem and subproblem are solved iteratively until the optimum solution is found. The Bender's decomposition method is used to solve an MIP problem with linking variables[3], but cannot be directly applied to the block structured problems with linking constraints, e.g., the large-scale MILP problems to be solved in this work.

Another technique for decomposing mixed-integer programming problem with block angular structure was proposed in Rana (1992). It can be thought as an extension of Dantzig-Wolfe decomposition for mixed-integer linear programming (MILP). Unfortunately, it was developed for handling linking constrains only involving continuous variables. As a result, it has limited value.

Lagrangian relaxation techniques are widely used to convert a large problem (e.g., an NLP or MIP problem) into two smaller and more tractable problems. This creates a master problem and a subproblem, by relaxing the "complicating" constraint set (Geoffrion, 1974; Bertsekas, 1995). In our coordination problem, the linking constraints are considered as complicating constraints. Based on the separability of the problem, the resulting subproblem often can be further divided into a group of smaller independent subproblems. For a block structured problem, through Lagrangian relaxation (LR), the linking constraints are relaxed by introducing Lagrange multipliers, and the overall optimization problem is divided into a group of subproblems. For a given set of Lagrange multipliers, the subproblems require less computation than the original problem. Thus, the solution of the original problem is done through a two-level iterative approach. When the subproblems can be solved individually at one level, the Lagrange multipliers are updated at the other level, which actually performs the coordination of subproblems.

Subgradient methods are frequently used to optimize dual functions in Lagrangian relaxation for separable MIP problems[4]. Subgradient optimization can be considered as an extension of a gradient method to the optimization of nondifferentiable functions. Lagrangian relaxation and subgradient optimization methods can be directly applied to solving a block structured MIP problem by taking advantage of the special structure. Based on the improvements made in its convergence performance, the subgradient optimization method and its variants have shown satisfactory computational performance in solving typical IP problems, such as the traveling salesman problem, manufacturing scheduling problem, and assignment problem (Fumero, 2001) and the job shop scheduling problems (Zhao *et al.*, 1999). Therefore, we have chosen Lagrangian relaxation and subgradient optimization methods as the basis for our MIP coordination study.

---

[3] In a decomposable large-scale system (problem), linking variables (or sometimes called complicating variables) are those variables associated with multiple subsystems (subproblems).

[4] This chapter does not intend to discuss NLP problems, although subgradient optimization methods can also be used to solve decomposable NLP problems

5.1.3 Subgradient Optimization Techniques

This section discusses the basic idea of Lagrangian relaxation and subgradient optimization. Lagrangian relaxation is used to convert a large-scale problem into smaller subproblems, while subgradient optimization methods are used to iteratively solve these subproblems to achieve the optimum solution.

Consider the following optimization problem:

$$\begin{aligned} &\text{(P)} \ \min \ f(\mathbf{x}) \\ &\text{subject to: } \mathbf{g}(\mathbf{x}) \le \mathbf{0} \\ &\mathbf{x} \in \mathcal{X} \end{aligned} \tag{5.1}$$

where $\mathbf{x} \in \mathcal{X}$ is a subset of R^n and $\mathbf{g} \in R^m$ is the set of complicating or linking constraints. $\mathcal{X}$ represents the feasible domain with respect to the other constraints that are not part of complicating constraints and may involve integer variables. Since $\mathbf{g}(\mathbf{x})$ contains the complicating (linking) constraint set, by applying Lagrangian relaxation, the original problem (P) has its Lagrangian dual given by:

$$\begin{aligned} &L(\boldsymbol{\lambda}) = \min \ f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}) \\ &\text{subject to: } \mathbf{x} \in \mathcal{X}, \ \boldsymbol{\lambda} \ge \mathbf{0} \end{aligned} \tag{5.2}$$

Then the Lagrangian dual problem is given as:

$$\text{(LD)} \ \max_{\boldsymbol{\lambda} \ge \mathbf{0}} \ L(\boldsymbol{\lambda}) \tag{5.3}$$

and the optimal solution is denoted as $L^* = L(\boldsymbol{\lambda}^*)$. The (LD) problem in (5.3) is nondifferentiable in general, e.g., the Lagrangian dual function $L(\boldsymbol{\lambda})$ of a linear integer programming problem is piece-wise linear, concave and continuous but non-smooth (Fumero, 2001). Then, the subgradient method is commonly used to maximize the dual function. It should be noted that the solution of original problem (P) is equivalent to the solution of problems in equations (5.2) and (5.3). The key issue is then to determine the Lagrange multipliers $\boldsymbol{\lambda}^*$ that correspond to the optimal solution $\mathbf{x}^*$.

In the subgradient optimization method, at any iteration k, given $\boldsymbol{\lambda}_k \geq \mathbf{0}$, the multipliers are updated by:

$$\boldsymbol{\lambda}_{k+1} = \mathbf{P}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}_k + t_k \mathbf{s}^k) \tag{5.4}$$

where $\mathbf{P}_{\boldsymbol{\lambda}}$ denotes a projection operator which is used to ensure $\boldsymbol{\lambda}_{k+1}$ is feasible. For example, when $\boldsymbol{\lambda}$ consists of non-negative elements (i.e., $\boldsymbol{\lambda} \geq \mathbf{0}$), the projection operator $\mathbf{P}_{\boldsymbol{\lambda}}$ can be chosen as $\max\{0, \lambda_i\}$, where λ_i is an element of vector $\boldsymbol{\lambda}$. Let $\partial L(\boldsymbol{\lambda}_k)$ denote the set of subgradients of L at $\boldsymbol{\lambda}_k$. Then $\mathbf{s}^k \in \partial L(\boldsymbol{\lambda}_k)$ represents a subgradient direction, while t_k is a suitable step size. The basis of the above scheme is based on the well known fact that, given a non-optimal feasible solution $\boldsymbol{\lambda}_k$ and $\mathbf{s}^k \in \partial L(\boldsymbol{\lambda}_k)$, there exists a step size t_k such that $\mathbf{P}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}_k + t_k \mathbf{s}^k)$ is closer, in the Euclidean norm sense, to an optimal solution to problem (LD) than $\boldsymbol{\lambda}_k$ (Bazaraa and Sherali, 1981). Despite the simplicity of the basic idea of subgradient optimization, behavior such as weak convergence properties and nonmonotonicity of the dual function was found in early implementations. Significant efforts have been made to improve its performance in recent two decades. Most of the research follows two streams: one is focused on selection of an appropriate step size t_k and the other is focused on determining an appropriate search direction $\mathbf{s}^k$.

5.1.4 Primal Solution Recovery

It must be noted that, the subgradient optimization approach provides an optimal solution to the LD problem, but does not solve the original problem (P) (5.1) in the primal space. Thus, it is necessary to use the optimal solution of problem (LD) to construct feasible solutions to the original problem (P). In the literature, Lagrangian heuristics (Beasley, 1993) are used to retrieve primal solutions from the LD solution.

Sherali and Choi (1996) provided primal solution recovery mechanisms when using subgradient optimization methods to solve Lagrangian duals of linear programs. Their work also discussed general difficulties in developing simple but efficient primal solution recovery heuristics. Barahona and Anbil (2000) developed the volume algorithm as an extension of deflected subgradient methods to generate primal solutions for a linear programming problem. This algorithm is based on the fact that a primal solution can be derived from the volumes below the faces, which are active at the maximum point of the dual function. A primal solution can be constructed as a convex combination of the solutions of subproblems; however, for integer programming problems, there is no guarantee of satisfying integrality from the convex combinations of points. Thus, this algorithm can be used to obtain a near-optimal integer solution in general.

In Beasley (1993), it is claimed that "Designing a Lagrangian heuristic for a particular LLBP[5] is an art, the success of which is judged solely by computational performance, i.e., whether a particular Lagrangian heuristic gives good quality (near-optimal or optimal) solutions in a reasonable computation time". Since these are heuristics and usually case dependent, there are no standard design procedures that can guarantee computational performance. The essential idea of designing Lagrangian heuristics is straightforward, and most heuristics attempt to make the best use of the solution of LD problem (e.g., a lower bound on the optimal solution to the original problem (P)) and the structural information of the solution to the LD problem (e.g., the decision variables having values close to an integer may be worthy of attention in the heuristic design).

5.2 Enhancements on Subgradient Algorithms

A modified subgradient algorithm for Lagrangian relaxation discussed in Fumero (2001) incorporates a number of "most promising" heuristics in the field of subgradient optimization. Those heuristics include the widely used step size update schemes (Poljak,

---

[5] Lagrangian lower bound program (Beasley, 1993), which is the linear case of the Lagrangian dual problem in (5.3)

1969; Held *et al.*, 1974), variable target value techniques by Kim *et al.* (1991) for updating the estimate of the optimal value of the dual function, and the search direction modification scheme proposed by Camerini *et al.* (1975). Good performance of the modified subgradient algorithm is reported based on computational experience in the Traveling Saleman and Assignment problems. In this work, a new heuristic to improve both search direction and step size determination is proposed. With the proposed heuristic, an extension is made to the subgradient algorithm given in Fumero (2001). In the improved algorithm, some recently developed heuristics are also incorporated, and a preliminary theoretical analysis shows the computational performance may be improved over existing algorithms.

5.2.1 Improved Subgradient Search Directions

A brief introduction is given for the most widely used heuristics in the literature, in terms of the two research streams to improve the subgradient optimization algorithms.

Step Size Selection

The most commonly used approach to step size calculation in subgradient optimization is:

$$t_k = \frac{\theta_k(\hat{L}_k - L(\boldsymbol{\lambda}_k))}{\|\mathbf{s}^k\|^2} \quad \text{with } 0 \le \theta_k \le 2 \tag{5.5}$$

where $\hat{L}_k$ is the estimation of L^* at current iteration k, and $L(\boldsymbol{\lambda}_k)$ is the objective function value of problem (LD) at iteration k. $\mathbf{s}^k$ is a modified subgradient direction which will be discussed later. The quality of the estimate $\hat{L}_k$ is very important in determining the step size. Based on the fact that the subgradient optimization algorithm will eventually drive $L(\boldsymbol{\lambda}_k)$ to L^*, the best objective value $L^c = \max_{j=1}^{k}\{L_j\}$ should converge to L^*. When the quality of an initial estimation (i.e., L^0) of L^* is not known (e.g., usually an overestimate of L^*), it is desired to shift the weights of $\hat{L}_k$ on L^0 to the best objective value L^c at the current iteration. A commonly implemented strategy is given in Bazaraa and Sherali (1981):

$$\hat{L}_k = \alpha_k L^0 + (1 - \alpha_k)L^c \tag{5.6}$$

where $\{\alpha_k\}$ is a properly designed sequence. It should be noted that, to avoid the frequent occurrence of $\hat{L}_k$ turning out to be an underestimate of L^*, the sequence of $\{\alpha_k\}$ should be appropriately designed. For instance, the sequence $\{\alpha_k\}$ weights L^0 heavily in the initial stages of the process when L^c is likely to be significantly less than L^*, and shifts the weight onto L^c gradually at first and then more rapidly later when L^c approaches L^*. One sequence of $\{\alpha_k\}$ which provides the desired features is given in Bazaraa and Sherali (1981):

$$\alpha_k = \begin{cases} e^{-0.6933(k/r_1)^{3.26}}, & \text{if } k < r_2; \\ \epsilon_0, & \text{if } k \geq r_2. \end{cases} \tag{5.7}$$

where r_1 is a tuning parameter by which we may delay, or hasten, the shift of the dominating weight from L^0 to L^c; ϵ_0 should take a suitable value as the minimum weight of L^0 when the solution is close to the optimum; and r_2 is further determined as the smallest integer k which satisfies:

$$\epsilon_0 \geq e^{-0.6933(k/r_1)^{3.26}} \tag{5.8}$$

The shape of the weighting function $\alpha(k)$ is shown in Figure 5.1.

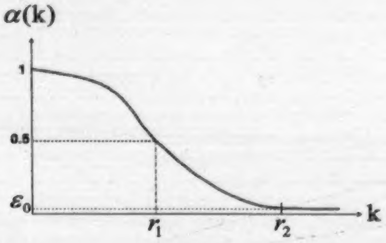


Figure 5.1: Weighting function dynamics

Typically, if L^0 is close to L^* (i.e., we have a good estimation of the optimum value), it is desired to stay close to L^0 and rely on the initial estimation for more iterations, thus a higher value of r_1 and a small ϵ_0 should be used. On the other hand, if L^0 is a poor estimate (i.e., far from L^*), it is desired to rely on L^c more heavily, thus a lower value of r_1 and a larger value of ϵ_0 should be chosen.

Search Direction Adjustment

Another stream of research has been focused on improving the subgradient optimization methods through subgradient direction modification techniques. Recall that the Lagrange multipliers can be updated by equation (5.4), where $\partial L(\boldsymbol{\lambda}_k)$ denotes the set of subgradients of L at $\boldsymbol{\lambda}_k$. The subgradient modification scheme proposed by Camerini *et al.* (1975) is one practical approach and employs a linear combination of the current subgradient $\boldsymbol{\mu}_k$ and last modified subgradient direction $\mathbf{s}^{k-1}$. The update formula is given by:

$$\mathbf{s}^k = \boldsymbol{\mu}^k + \beta_k \mathbf{s}^{k-1} \tag{5.9}$$

where $\boldsymbol{\mu}^k = \partial L(\boldsymbol{\lambda}_k)$ and β_k is a suitable scalar ($\mathbf{s}^{k-1} = 0$ when $k = 0$). The subgradient $\boldsymbol{\mu}^k$ is evaluated for a given multiplier $\boldsymbol{\lambda}_k$. Generally, for an optimization problem in (5.1), the subgradient $\boldsymbol{\mu}^k(\boldsymbol{\lambda}_k) = \mathbf{g}(\mathbf{x}; \boldsymbol{\lambda}_k)$, where $\mathbf{x}(\boldsymbol{\lambda}_k) = arg\min_{\mathbf{x}\in\mathcal{X}}\{f(\mathbf{x}) + \boldsymbol{\lambda}_k^T\mathbf{g}(\mathbf{x})\}$. It should be noted that formula (5.9) is in fact equivalent to using all preceding subgradient direction information. Before further discussion, let us define $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$ as the vector connecting $\boldsymbol{\lambda}_k$ to $\boldsymbol{\lambda}^*$ in an Euclidean space.

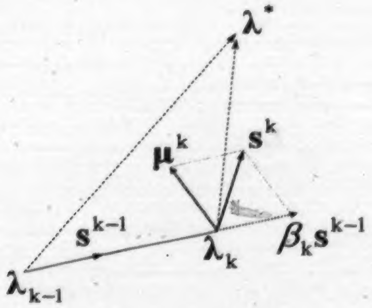


Figure 5.2: Geometric interpretation - obtuse angle

As shown in Figure 5.2, $\mathbf{s}^{k-1}$ is the direction from vector $\boldsymbol{\lambda}_{k-1}$ (the previous multiplier) to vector $\boldsymbol{\lambda}_k$ (the current multiplier). At current iteration k, the subgradient $\boldsymbol{\mu}^k = \partial L(\boldsymbol{\lambda}_k)$. The fact is that, based on the update formula (5.9), the angle spanned by $\mathbf{s}^{k-1}$ and the direction of $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$ is an acute angle ($< \frac{\pi}{2}$). $\boldsymbol{\lambda}_k$ is generated from $\boldsymbol{\lambda}_{k-1}$ following direction $\mathbf{s}^{k-1}$. Thus, when $\boldsymbol{\mu}^k$ and $\mathbf{s}^{k-1}$ form an obtuse angle ($> \frac{\pi}{2}$), i.e., when $\mathbf{s}^{k-1} \cdot \boldsymbol{\mu}^k < 0$, it is desired to adjust the direction of subgradient $\boldsymbol{\mu}^k$ closer to $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$. Intuitively, if we choose a β_k as a sufficiently small positive number, the resulting vector $\mathbf{s}^k$ forms a smaller angle with the direction of $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$ than does $\boldsymbol{\mu}^k$.

In Camerini *et al.* (1975), the choice of β_k is given by:

$$\beta_k = \begin{cases} -\gamma_k \frac{\mathbf{s}^{k-1} \cdot \boldsymbol{\mu}^k}{\|\mathbf{s}^{k-1}\|^2}, & \text{if } \mathbf{s}^{k-1} \cdot \boldsymbol{\mu}^k < 0; \\ 0, & \text{otherwise.} \end{cases} \tag{5.10}$$

with $0 \leq \gamma_k \leq 2$. The scheme in equation (5.10) was proposed to determine the weighting factor β_k in the case the angle between current gradient direction $\boldsymbol{\mu}^k$ and previous modified direction $\mathbf{s}^{k-1}$ is an obtuse angle, i.e., $\mathbf{s}^{k-1} \cdot \boldsymbol{\mu}^k < 0$ in formula (5.10). In Figure 5.2, the optimal multiplier $\boldsymbol{\lambda}^*$ is used just for the purposes of geometric interpretation; however, it is not required in the calculation of β_k in the implementation of algorithms. The heuristic in equation (5.10) gives a quantitative method to obtain a proper weighting factor β_k, which guarantees the resulting direction $\mathbf{s}^k$ is closer to the direction of $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$.

Like other heuristics, this scheme also introduces a scalar tuning parameter γ_k. The choice of γ_k may depend on the nature of problems to be solved, and for different choice of γ_k the algorithm presents different computational efficiency. In Camerini *et al.* (1975), the value of γ_k is determined by:

$$\gamma_k = \frac{\rho + |\cos\alpha|}{|\cos\alpha|(1 + \rho|\cos\alpha|)} \tag{5.11}$$

where α is the angle formed by $\mathbf{s}^{k-1}$ and $\boldsymbol{\mu}^k$; $\rho = \frac{\cos\phi}{\cos\psi}$, where ϕ and ψ stand for the angles which $\mathbf{s}^{k-1}$ and $\boldsymbol{\mu}^k$ formed with vector $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$, respectively. Geometrically, ρ indicates the relative location of vector $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$ with respect to $\mathbf{s}^{k-1}$ and $\boldsymbol{\mu}^k$. In practice, the vector $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$ is not known, so an estimate of ρ has to be made by trial and error. For example, a simple heuristic estimation of ρ is $\rho = 1$, which assumes that on the average, $\mathbf{s}^{k-1}$ and $\boldsymbol{\mu}^k$ form equal angles with respect to $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$, and this corresponds to a value of $\gamma_k = \frac{1}{|\cos\alpha|}$. An assumption often used is $\alpha = \frac{3\pi}{4}$, resulting in $\gamma_k = \sqrt{2}$, for which good computational experience has been reported in many applications (Camerini *et al.*, 1975). In addition, a fixed $\gamma = 1$ would amount to using a direction orthogonal to $\mathbf{s}^{k-1}$. With an appropriate γ, this modification can significantly improve the convergence of subgradient optimization methods. A large amount of computational experience has shown its value in improving the convergent rate of subgradient optimization; moreover, the policy (5.10) tends to alleviate "zig-zag" behavior of the sequence of $\{\boldsymbol{\lambda}_k\}$ (Camerini *et al.*, 1975).

Adjustment For A Feasible Subgradient Direction

In the context of Lagrangian relaxation, another approach to solving the dual problem (LD) is steepest ascent optimization (Bazaraa and Goode, 1979). There, research has focused on finding a feasible direction of steepest ascent, which improves the dual objective function monotonically iteration by iteration. Although our work focuses on subgradient optimization, the concept of *a feasible direction* in steepest ascent optimization should still be valid for the purpose of our work.

Definition (Bazaraa and Goode, 1979). Let L be concave and consider the problem in (5.3). When $\boldsymbol{\lambda} \geq \mathbf{0}$, s is called a *feasible direction* for L at $\boldsymbol{\lambda}$ if $\lambda_i = 0$ implies that $s_i \geq 0$, where $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, ..., \lambda_m]$ and $\mathbf{s} = [s_1, s_2, ..., s_m]$.

To understand this definition, consider the following fact. In ascent optimization, if $\boldsymbol{\lambda}_k > \mathbf{0}$, and if we move along an ascent direction $\mathbf{s}_a^k$ for $L(\boldsymbol{\lambda}_k)$, then there exists a $\delta > 0$ such that $L(\boldsymbol{\lambda}_k + t_k\mathbf{s}_a^k) > L(\boldsymbol{\lambda})$ and $\boldsymbol{\lambda}_k + t_k\mathbf{s}_a^k$ remains feasible for all $t_k \in [0, \delta]$. On the other hand, if at least one component of $\boldsymbol{\lambda}_k$ is equal to zero, moving along the direction $\mathbf{s}_a^k$ may destroy feasibility for any step size t_k (Bazaraa and Goode, 1979). The same concept also applies to the subgradient optimization scheme discussed in previous sections. When the Lagrange multiplier $\boldsymbol{\lambda}$ is updated through equation (5.4), which directly uses the search direction $\mathbf{s}^k$ obtained through equation (5.9). It should be noted that, the search direction $\mathbf{s}^k$ may not be a feasible direction for updating the Lagrange multiplier $\boldsymbol{\lambda}$. This may worsen the zig-zagging behaviour of subgradient optimization methods.

To incorporate the concept of a "feasible direction", a heuristic is developed as follows by modifying the projection discussed as Theorem 2 in Bazaraa and Goode (1979). For the Lagrangian dual function $L(\boldsymbol{\lambda}_k)$ defined in problem (5.3), if a subgradient direction $\mathbf{s}^k \in \partial L(\boldsymbol{\lambda}_k)$, its i^{th} component s_i^k is projected as follows:

$$\hat{s}_i^k = \begin{cases} s_i^k, & \text{if } i \notin I_k(\boldsymbol{\lambda}); \\ 0, & \text{otherwise.} \end{cases} \tag{5.12}$$

where $I_k(\boldsymbol{\lambda})$ is an index set determined by:

$$I_k(\boldsymbol{\lambda}) = \{i \mid s_i^k \leq 0 \text{ and } \lambda_k^i = 0\} \tag{5.13}$$

Then, the direction $\hat{\mathbf{s}}^k = [\hat{s}_1^k, \hat{s}_2^k, ..., \hat{s}_m^k]$ is a feasible direction. Let us denote the projection operation as $\hat{\mathbf{s}}^k = \mathbf{P}_{\mathbf{s}}(\mathbf{s}^k)$. In brief, the proposed heuristic, which is described by (5.10) and (5.12), has the following properties:

1. Putting more emphasis on determining a feasible subgradient direction;
2. Yielding a direction $\hat{\mathbf{s}}^k$ not worse than $\mathbf{s}^k$;
3. Guaranteeing convergence of multiplier $\boldsymbol{\lambda}$ in an Euclidean norm sense;
4. Providing a larger range of step size that can improve the convergence of the multipliers.

Next, we are going to show some favorable properties of the resulting direction $\hat{\mathbf{s}}^k$.

In addition to the feasibility of the direction, as is shown in Figure 5.3, when we try to move from $\boldsymbol{\lambda}_k$ to $\boldsymbol{\lambda}^*$, it is obvious that the direction $\hat{\mathbf{s}}^k$ is a direction at least as good as, if not better than, the direction $\mathbf{s}^k$, because the projection of $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$ on $\hat{\mathbf{s}}^k$ (i.e., $\frac{(\boldsymbol{\lambda}^*-\boldsymbol{\lambda}_k)^T\hat{\mathbf{s}}^k}{\|\hat{\mathbf{s}}^k\|}$) is at least as large as, if not greater than, that on $\mathbf{s}^k$ (i.e., $\frac{(\boldsymbol{\lambda}^*-\boldsymbol{\lambda}_k)^T\mathbf{s}^k}{\|\mathbf{s}^k\|}$).

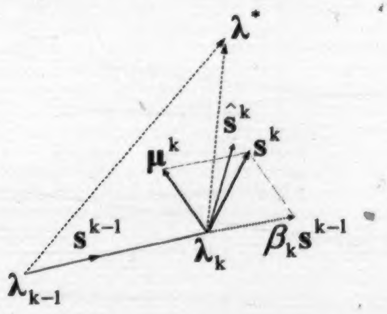


Figure 5.3: Adjustment of subgradient direction

Next, we are going to show that $\hat{\mathbf{s}}^k$ is a subgradient direction at least as good as $\mathbf{s}^k$. Based on the fact that $\lambda_i^* s_i^k \leq 0$ for all $i \in I_k$, we have

$$\begin{aligned}(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \mathbf{s}^k &= \sum_{i \notin I_k} (\lambda_i^* - \lambda_k^i) s_i^k + \sum_{i \in I_k} \lambda_i^* s_i^k \\ &\leq \sum_{i \notin I_k} (\lambda_i^* - \lambda_k^i) s_i^k = (\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \hat{\mathbf{s}}^k \end{aligned} \tag{5.14}$$

In addition, from the definition of Euclidean norm (Bertsekas, 1995), we can easily obtain that:

$$\|\mathbf{s}^k\| \geq \|\hat{\mathbf{s}}^k\| \tag{5.15}$$

Hence, assuming that $\hat{\mathbf{s}}^k$ and $\mathbf{s}^k$ are non-zero vectors, the following inequality holds:

$$\frac{(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \mathbf{s}^k}{\|\mathbf{s}^k\|} \leq \frac{(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \hat{\mathbf{s}}^k}{\|\hat{\mathbf{s}}^k\|} \tag{5.16}$$

In other words, referring to Figure 5.3, the vector $\boldsymbol{\lambda}_k \boldsymbol{\lambda}^*$ can have a larger projection onto the vector $\hat{\mathbf{s}}^k$, which means the modified direction $\hat{\mathbf{s}}^k$ is at least as good a direction as $\mathbf{s}^k$.

Next, we show the convergence of the subgradient optimization algorithm based on the improved subgradient direction. In other words, if the Lagrange multiplier update strategy is given by :

$$\boldsymbol{\lambda}_{k+1} = \mathbf{P}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}_k + t_k \hat{\mathbf{s}}^k) \tag{5.17}$$

where $\mathbf{P}_{\boldsymbol{\lambda}}$ is a projection operator on the closed convex set $M = \{\boldsymbol{\lambda} \geq \mathbf{0}, L(\boldsymbol{\lambda}) > -\infty\}$ (Bertsekas, 1995), and if the step size is chosen as (Camerini *et al.*, 1975):

$$0 < t_k \leq \frac{L(\boldsymbol{\lambda}^*) - L(\boldsymbol{\lambda}_k)}{\|\hat{\mathbf{s}}^k\|^2} \tag{5.18}$$

we should have $\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k\| > \|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_{k+1}\|$.

Then, we have:

$$\left\|\boldsymbol{\lambda}_k + t_k \hat{\mathbf{s}}^k - \boldsymbol{\lambda}^*\right\|^2 = \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}^*\|^2 - 2t_k(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \hat{\mathbf{s}}^k + t_k^2 \left\|\hat{\mathbf{s}}^k\right\|^2 \tag{5.19}$$

Lemma 1 in Camerini *et al.* (1975), which is based on the definition of subgradient, gives the inequality:

$$(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \boldsymbol{\mu}^k \geq L(\boldsymbol{\lambda}^*) - L(\boldsymbol{\lambda}_k) \tag{5.20}$$

Furthermore, *Lemma 2* in Camerini *et al.* (1975) states that, for a given direction $\mathbf{s}^k$ updated by the scheme in equations (5.9) and (5.10), the inequality:

$$(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \mathbf{s}^k \geq (\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \boldsymbol{\mu}^k \tag{5.21}$$

holds. Therefore, from inequalities (5.20) and (5.21), we obtain:

$$(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \mathbf{s}^k \geq L(\boldsymbol{\lambda}^*) - L(\boldsymbol{\lambda}_k) \tag{5.22}$$

and combine (5.22) with (5.14), we have:

$$(\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k)^T \hat{\mathbf{s}}^k \geq L(\boldsymbol{\lambda}^*) - L(\boldsymbol{\lambda}_k) \tag{5.23}$$

Then, from (5.19) and (5.23), we can get:

$$\left\|\boldsymbol{\lambda}_k + t_k \hat{\mathbf{s}}^k - \boldsymbol{\lambda}^*\right\|^2 \leq \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}^*\|^2 - 2t_k(L(\boldsymbol{\lambda}^*) - L(\boldsymbol{\lambda}_k)) + t_k^2 \left\|\hat{\mathbf{s}}^k\right\|^2 \tag{5.24}$$

It is straightforward to verify that, for the step size t_k that satisfies (5.18), the sum of the last two terms in (5.24) is negative, so the above inequality yields:

$$\left\|\boldsymbol{\lambda}_k + t_k \hat{\mathbf{s}}^k - \boldsymbol{\lambda}^*\right\| < \|\boldsymbol{\lambda}_k - \boldsymbol{\lambda}^*\| \tag{5.25}$$

Because $\boldsymbol{\lambda}^* \in M$ and the projection operation $\mathbf{P}_{\boldsymbol{\lambda}}$ is non-expansive[6], we have:

$$\left\|\mathbf{P}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}_k + t_k \hat{\mathbf{s}}^k) - \boldsymbol{\lambda}^*\right\| \leq \left\|\boldsymbol{\lambda}_k + t_k \hat{\mathbf{s}}^k - \boldsymbol{\lambda}^*\right\| \tag{5.26}$$

By combining the last two inequalities, the property $\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_k\| > \|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_{k+1}\|$ has been proved.

Moreover, for step size selection, if we adopt the formula given in (5.18), because $\left\|\mathbf{s}^k\right\| \geq \left\|\hat{\mathbf{s}}^k\right\|$, a larger range of step size t_k can be chosen due to the use of the improved direction $\hat{\mathbf{s}}^k$.

For the proposed search direction adjustment scheme, in particular, properties 2 and 4 may significantly improve the rate of convergence of the subgradient algorithms, while property 1 may alleviate the zig-zagging behavior of the algorithms.

---

[6] The mapping $f : R^n \to \mathcal{X}$ defined by $f(x) = [x]^+$ is continuous and non-expansive, that is: $\|[x]^+ - [y]^+\| \leq \|x - y\| \ \forall x, y \in R^n$ (Bertsekas, 1995).

5.2.2 An Enhanced Subgradient Algorithm

By incorporating the proposed extensions, further enhancement is expected to the subgradient optimization algorithm proposed in Fumero (2001). A decentralized version of the modified algorithm can also be developed for separable large-scale optimization problems (MILP, MIQP, MINLP), which takes advantage of the special problem structure.

A Version for Centralized Computing Environments

Next, an algorithmic scheme of the enhanced subgradient optimization algorithm is given as follows for an implementation on a centralized computing platform. Moreover, to help explain the proposed subgradient optimization algorithm, the diagrams in Figures 5.4 and 5.5 give a flowchart description of the implementation.

- Initialization. Choose a starting solution $\boldsymbol{\lambda}_1 \in M = \{\boldsymbol{\lambda} \geq \mathbf{0}, L(\boldsymbol{\lambda}) > -\infty\}$ and compute $L(\boldsymbol{\lambda}_1)$. Determine $\boldsymbol{\mu}_1 \in \partial L(\boldsymbol{\lambda}_1)$. Let $\hat{\mathbf{s}}^1 = \boldsymbol{\mu}_1$, and if $\|\hat{\mathbf{s}}^1\| = 0$, STOP with $\boldsymbol{\lambda}_1$ as an optimal solution to the problem (LD). Otherwise, choose an initial estimation $L^0 > L^* = \sup\{L(\boldsymbol{\lambda}) : \boldsymbol{\lambda} \in M\}$ and set $\hat{L}_1 = L^0$ as the target value for calculating the stepsize. Let $(\boldsymbol{\lambda}^c, \hat{\mathbf{s}}^c, L^c) = (\boldsymbol{\lambda}_1, \hat{\mathbf{s}}^1, L(\boldsymbol{\lambda}_1))$. Select appropriate positive values for ϵ_0 and ϵ, and select appropriate positive integral values for $\bar{\nu}$, $\bar{\theta}$ and r_1 as tuning parameters. Let $\theta_1 = 1, r = 0, \nu = 0, k = 1$, and $\alpha_0 = 1$ and go to Step 1.
- Step 1. Given $\boldsymbol{\lambda}_k$, β_k ($\beta_1 = 0$), $L(\boldsymbol{\lambda}_k)$, $\hat{\mathbf{s}}^k$ and $\hat{L}_k$, determine the stepsize:

  $$t_k = \frac{1}{\theta_k}\left[\frac{\hat{L}_k - L(\boldsymbol{\lambda}_k)}{\|\hat{\mathbf{s}}^k\|^2}\right] \tag{5.27}$$

  Update the multipliers with $\boldsymbol{\lambda}_{k+1} = \mathbf{P}_{\boldsymbol{\lambda}}(\boldsymbol{\lambda}_k + t_k \hat{\mathbf{s}}^k)$. Compute $L(\boldsymbol{\lambda}_{k+1})$ and determine $\boldsymbol{\mu}_{k+1} \in \partial L(\boldsymbol{\lambda}_{k+1})$. If $\left\|\boldsymbol{\mu}^{k+1}\right\| = 0$, terminate with $\boldsymbol{\lambda}_{k+1}$ as an optimal solution to the problem. Otherwise, replace k by $k+1$. If k reaches the prespecified maximum iteration number $\bar{k}$, STOP; else, go to Step 2.
- Step 2. If $\hat{\mathbf{s}}^{k-1} \cdot \boldsymbol{\mu}^k < 0$, set $\beta_k = -\gamma_k \frac{\hat{\mathbf{s}}^{k-1} \cdot \boldsymbol{\mu}^k}{\|\hat{\mathbf{s}}^{k-1}\|^2}$, and perform the projection $\hat{\mathbf{s}}^k = \mathbf{P}_{\mathbf{s}}(\boldsymbol{\mu}^k + \beta_k \hat{\mathbf{s}}^{k-1})$; otherwise, $\hat{\mathbf{s}}^k = \mathbf{P}_{\mathbf{s}}(\boldsymbol{\mu}^k)$. Then, go to Step 3 if $r < r_2$ or to Step 6 if $r = r_2$.

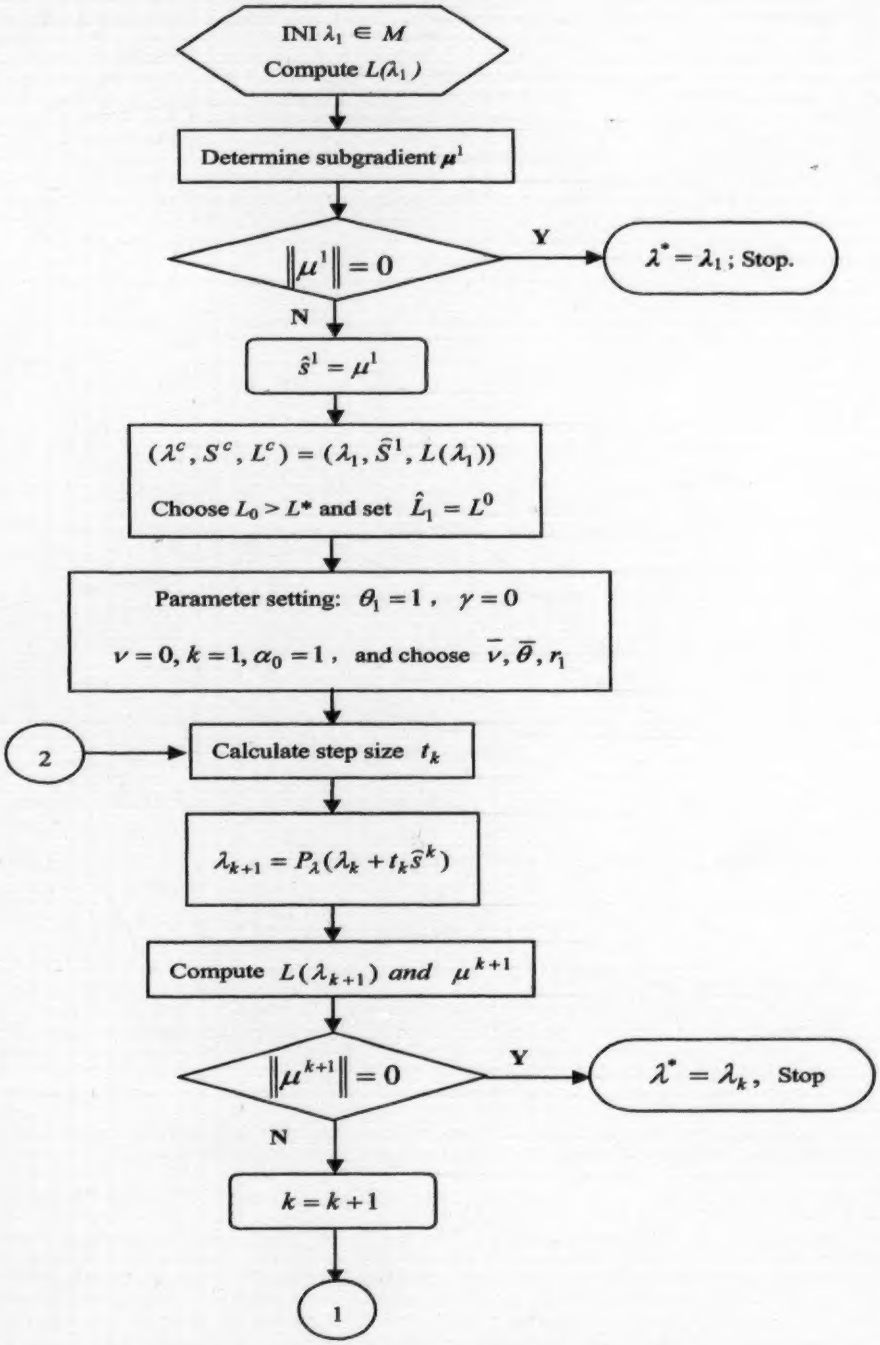

INI $\lambda_1 \in M$
Compute $L(\lambda_1)$
Determine subgradient μ^1
$\|\mu^1\| = 0$
Y
$\lambda^* = \lambda_1$; Stop.
N
$\hat{s}^1 = \mu^1$
$(\lambda^c, S^c, L^c) = (\lambda_1, \hat{S}^1, L(\lambda_1))$
Choose $L_0 > L^*$ and set $\hat{L}_1 = L^0$
Parameter setting: $\theta_1 = 1$, $\gamma = 0$
$\nu = 0, k = 1, \alpha_0 = 1$, and choose $\bar{\nu}, \bar{\theta}, r_1$
2
Calculate step size t_k
$\lambda_{k+1} = P_\lambda(\lambda_k + t_k \hat{s}^k)$
Compute $L(\lambda_{k+1})$ and μ^{k+1}
$\|\mu^{k+1}\| = 0$
Y
$\lambda^* = \lambda_k$, Stop
N
$k = k + 1$
1

Figure 5.4: Improved subgradient algorithm - part I

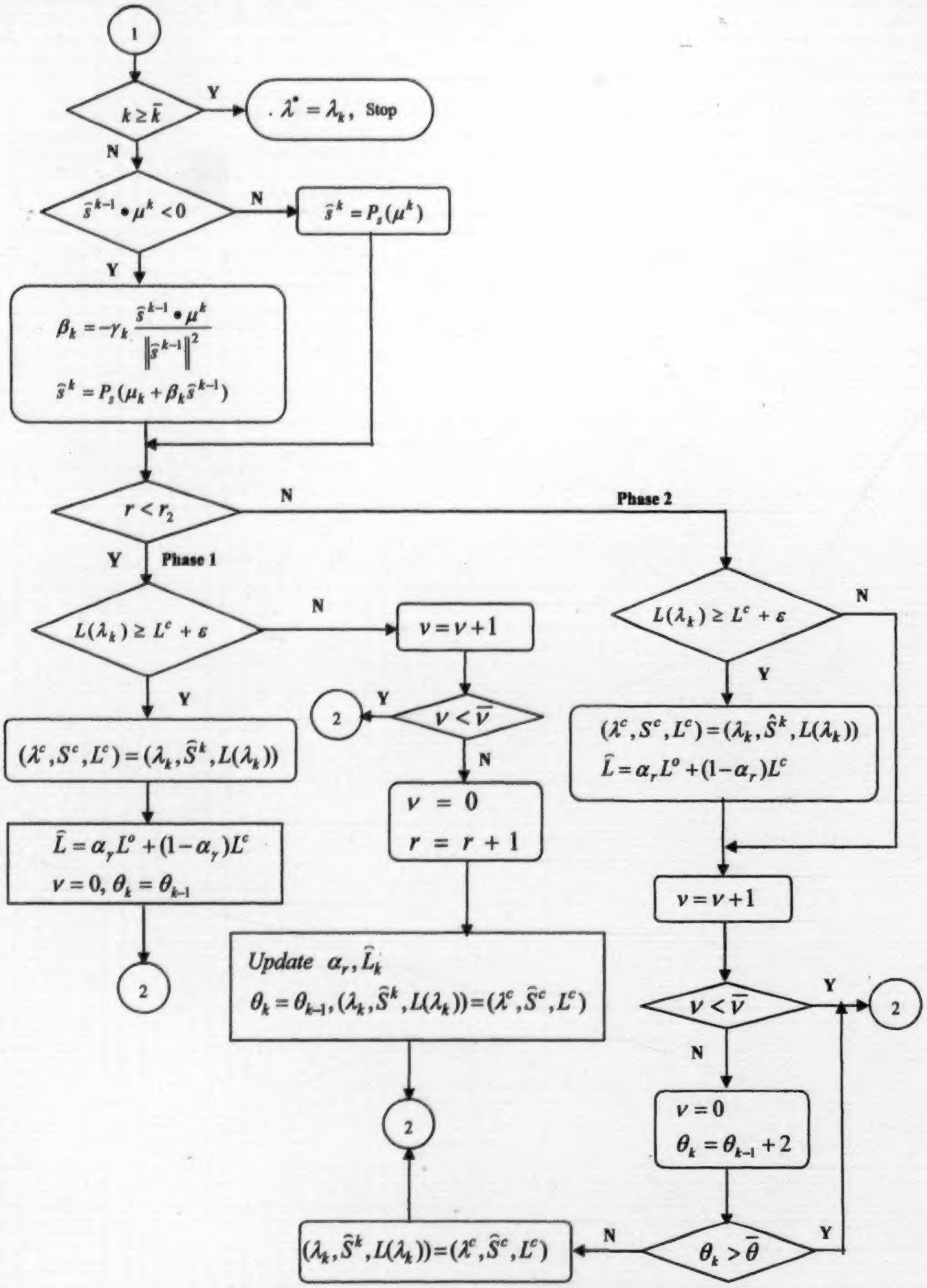

1
$k \geq \bar{k}$
Y
$.\ \lambda^* = \lambda_k$, Stop
N
$\hat{s}^{k-1} \bullet \mu^k < 0$
N
$\hat{s}^k = P_s(\mu^k)$
Y
$\beta_k = -\gamma_k \frac{\hat{s}^{k-1} \bullet \mu^k}{\|\hat{s}^{k-1}\|^2}$
$\hat{s}^k = P_s(\mu_k + \beta_k \hat{s}^{k-1})$
$r < r_2$
N
Phase 2
Y
Phase 1
$L(\lambda_k) \geq L^c + \varepsilon$
N
$\nu = \nu + 1$
Y
2
Y
$\nu < \bar{\nu}$
N
$(\lambda^c, S^c, L^c) = (\lambda_k, \hat{S}^k, L(\lambda_k))$
$\nu = 0$
$r = r + 1$
$\hat{L} = \alpha_r L^o + (1 - \alpha_r) L^c$
$\nu = 0,\ \theta_k = \theta_{k-1}$
2
Update $\alpha_r, \hat{L}_k$
$\theta_k = \theta_{k-1}, (\lambda_k, \hat{S}^k, L(\lambda_k)) = (\lambda^c, \hat{S}^c, L^c)$
2
$L(\lambda_k) \geq L^c + \varepsilon$
N
Y
$(\lambda^c, S^c, L^c) = (\lambda_k, \hat{S}^k, L(\lambda_k))$
$\hat{L} = \alpha_r L^o + (1 - \alpha_r) L^c$
$\nu = \nu + 1$
$\nu < \bar{\nu}$
Y
2
N
$\nu = 0$
$\theta_k = \theta_{k-1} + 2$
$(\lambda_k, \hat{S}^k, L(\lambda_k)) = (\lambda^c, \hat{S}^c, L^c)$
N
$\theta_k > \bar{\theta}$
Y

Figure 5.5: Improved subgradient algorithm - part II

Phase I:

- Step 3. If $L(\boldsymbol{\lambda}_k) \geq L^c + \epsilon$, go to Step 5; otherwise, go to Step 4.

- Step 4. Let $\nu = \nu + 1$. If $\nu < \bar{\nu}$, go to Step 1; otherwise (if $\nu = \bar{\nu}$), set $\nu = 0$ and replace r by $r + 1$. Compute $\alpha_r = e^{-0.6933(r/r_1))^{3.26}}$ and set $\hat{L}_k = \alpha_r L^0 + (1 - \alpha_r)L^c$. Let $\theta_k = \theta_{k-1}$, and reset $(\boldsymbol{\lambda}_k, \hat{\mathbf{s}}^k, L(\boldsymbol{\lambda}_k)) = (\boldsymbol{\lambda}^c, \hat{\mathbf{s}}^c, L^c)$. And go to Step 1.

- Step 5. Let $(\boldsymbol{\lambda}^c, \hat{\mathbf{s}}^c, L^c) = (\boldsymbol{\lambda}_k, \hat{\mathbf{s}}^k, L(\boldsymbol{\lambda}_k))$, and compute $\hat{L}_k = \alpha_r L^0 + (1 - \alpha_r)L^c$. Set $\nu = 0$, $\theta_k = \theta_{k-1}$, and go to Step 1.

Phase II:

- Step 6. If $L(\boldsymbol{\lambda}_k) \geq L^c + \epsilon$, go to Step 7; otherwise, go to Step 8.

- Step 7. Let $(\boldsymbol{\lambda}^c, \hat{\mathbf{s}}^c, L^c) = (\boldsymbol{\lambda}_k, \hat{\mathbf{s}}^k, L(\boldsymbol{\lambda}_k))$. Compute $\hat{L}_k = \alpha_r L^0 + (1 - \alpha_r)L^c$, and go to Step 8.

- Step 8. Replace ν by $\nu + 1$. If $\nu < \bar{\nu}$, go to Step 1; otherwise (if $\nu = \bar{\nu}$), set $\nu = 0$, and let $\theta_k = \theta_{k-1} + 2$. If $\theta_k \geq \bar{\theta}$, return to Step 1; otherwise, let $(\boldsymbol{\lambda}_k, \hat{\mathbf{s}}^k, L(\boldsymbol{\lambda}_k)) = (\boldsymbol{\lambda}^c, \hat{\mathbf{s}}^c, L^c)$ and return to Step 1.

A Version for Distributed Computing Environments

This section gives an algorithmic description of a decentralized version of the proposed subgradient optimization algorithm for solving large-scale MIP problems with decomposable structure. Consider a block-structured optimization problem:

$$
\begin{aligned}
\text{(P1)}\quad \min \quad & \sum_{i=1}^{p} f_i(\mathbf{x}_i) \\
\text{subject to:}\quad & \sum_{i=1}^{p} \mathbf{g}_i(\mathbf{x}_i) \leq \mathbf{0} \\
& \mathbf{x}_i \in \mathcal{X}_i
\end{aligned}
\tag{5.28}
$$

where $\mathbf{x}_i \in \mathcal{X}_i$ is a subset of R^n, $\mathbf{g}_i \in R^m$. $\mathcal{X}_i$ represents the individual feasible domain with respect to the local constraints of potential subproblems. Assume $\sum_{i=1}^{p} \mathbf{g}_i(\mathbf{x}_i)$

contains a complicating constraint set, by applying Lagrangian relaxation, the original problem (P1) has its Lagrangian dual given by:

$$L(\boldsymbol{\lambda}) = \min \sum_{i=1}^{p} f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^T \sum_{i=1}^{p} \mathbf{g}_i(\mathbf{x}_i)$$

$$\text{subject to: } \mathbf{x}_i \in \mathcal{X}_i, \ \boldsymbol{\lambda} \geq \mathbf{0} \tag{5.29}$$

Rewriting the Lagrangian function in (5.29), a Lagrangian function with separable structure is obtained:

$$\begin{aligned} L(\boldsymbol{\lambda}) &= \min \sum_{i=1}^{p} (f_i(\mathbf{x}_i) + \boldsymbol{\lambda}^T \mathbf{g}_i(\mathbf{x}_i)) \\ &= \sum_{i=1}^{p} L_i(\boldsymbol{\lambda}) \end{aligned} \tag{5.30}$$

Then, the Lagrangian dual (LD1) problem is given as:

$$\text{(LD1)} \ \max_{\boldsymbol{\lambda}_i \geq 0} \sum_{i=1}^{p} L_i(\boldsymbol{\lambda}) \tag{5.31}$$

In this case, the solution of the (LD1) problem can be achieved by solving p individual subproblems. The centralized version of the improved subgradient algorithm discussed in previous section can be accordingly modified to take advantage of a distributed computing environment.

In the **Initialization** step and **Step 1** of the centralized version, the dual function value $L(\boldsymbol{\lambda}_k)$ is evaluated by solving an LD problem as a whole; however, when the LD problem takes the form of (LD1), for a given multiplier $\boldsymbol{\lambda}_k$, the dual function of a subproblem $L_i(\boldsymbol{\lambda}_k)$ is evaluated at the solution of $\mathbf{x}_i(\boldsymbol{\lambda}_k) = arg \min_{\mathbf{x}_i \in \mathcal{X}_i} \{f(\mathbf{x}_i) + \boldsymbol{\lambda}_k^T \mathbf{g}_i(\mathbf{x}_i)\}$, which can be solved independently on a distributed computing platform.

In this case, the distributed version of the improved subgradient algorithm can be implemented in a distributed computing environment, where p computing nodes and an additional coordinator node are available. In brief, the evaluation of the p Lagrangian dual functions $L_i(\boldsymbol{\lambda}_k)$ is performed at the p computing nodes, respectively; while other computations are conducted at the coordinator computing node.

5.2.3 Discussions

By following different paths, similar heuristics for subgradient direction adjustment, as described in formula (5.12), are developed in Wang (2003) and Guta (2003).

Wang (2003) proposes a similar heuristic to equation (5.12); however, the heuristic developed in his work is only used for step size selection, i.e., the projected search direction $\hat{\mathbf{s}}^k$ is used to determine β_k in equation (5.10). In this thesis, the resulting search direction $\hat{\mathbf{s}}^k$ is also applied to equation (5.9) for subgradient search direction adjustment. In this case, the heuristic for search direction projection developed in this thesis extends the work by Wang (2003) to more general applications.

The mechanism behind many heuristics, which tend to improve the computational performance of subgradient optimization algorithms, has been discussed recently in Guta (2003). Most of the heuristics in the literature were developed to alleviate two kinds of zigzagging behavior of subgradient methods. The first kind of zigzagging behavior occurs when the subgradient direction $\boldsymbol{\mu}^k$ forms an obtuse angle with the previous direction $\mathbf{s}^{k-1}$, as shown in Figure 5.2; while the second kind of zigzagging is due to the location of k^{th} solution, i.e., even when the angle between $\boldsymbol{\mu}^k$ and $\mathbf{s}^{k-1}$ is acute, a movement of any size along $\mathbf{s}^k$ will cause an infeasible solution. A more rigorous definition of the two kinds of zigzagging can be found in Chapter 3 of Guta (2003). These zigzagging phenomena may significantly slow the convergence of subgradient optimization algorithms. The heuristics in the algorithm, which is developed in §5.2.2, actually intend to alleviate both kinds of zigzagging behavior and thus enhance the convergent speed of subgradient optimization.

Although the search direction adjustment strategy proposed in Guta (2003) is similar to the heuristic in equation (5.12) of this thesis, there are significant differences between these two pieces of work. Firstly, the work of Guta (2003) is derived from the concept of conditional subgradient methods (Larsson *et al.*, 1996); while the proposed scheme in this thesis is based on the concept of feasible direction in steepest ascent optimization framework. These are two independent pieces of work sharing the similar idea of maintaining the feasibility of search direction. Secondly, it should be noted that the design and implementation of the subgradient algorithms are quite different. In our work, the algorithm described in §5.2.2 has incorporated the most promising heuristics in the literature, including the proposed search direction adjustment heuristics (i.e., equation (5.12)) and the step size selection heuristics where variable target methods have been applied (i.e., equations (5.6) and (5.7)). In addition, a distributed version of the subgradient algorithm has been developed in this work as well.

5.3 Complexity Study

In the literature, mixed-integer linear programming (MILP) problems represent one important class of optimization problems, which have been mentioned in §5.1 for planning, scheduling, and resource allocation, *etc.* In addition, hybrid MPC applications, which involve integer variables, have been discussed in Bemporad and Morari (1999). It may be beneficial to apply an MILP-based target calculation to the hybrid MPC system to help achieve plant-wide optimal operations. Thus, MILP/ILP problems are the main focus for the remainder of this chapter.

In this section, a brief discussion on the theoretical complexity of the subgradient optimization algorithms will be given. Then a comprehensive empirical study is performed to gain insight into the scaling behavior of the enhanced subgradient algorithm.

We consider a large-scale block-angular mixed-integer linear programming problem with p subproblems:

$$\begin{aligned}
&\max \quad \sum_i \mathbf{c}_i^T \mathbf{x}_i \\
&\text{subject to} \\
&\sum_i \mathbf{A}_i \mathbf{x}_i \leq \mathbf{b}_0 \\
&\mathbf{B}_i \mathbf{x}_i \leq \mathbf{b}_i \\
&\mathbf{x}_i \geq \mathbf{0} \quad i = 1, 2, ..., p \\
&x_{ij} \in Z \quad \text{for } j \in I_i
\end{aligned} \tag{5.32}$$

where vectors $\mathbf{x}_i$ ($n_i \times 1$), $\mathbf{b}_i$ ($m_i \times 1$), $\mathbf{b}_0$ ($m_0 \times 1$), $\mathbf{c}_i$ ($n_i \times 1$), and matrices $\mathbf{A}_i$ ($m_0 \times n_i$), $\mathbf{B}_i$ ($m_i \times n_i$) are specific to subproblem "i". It should be noted that the set I contains the indices of integer variables in the decision vector $\mathbf{x}_i$.

5.3.1 Theoretical Analysis

In the literature, although it has been shown that some of the integer programming problems (e.g., the Uncapacitated Lot-Sizing Problem, the Shortest Path Problem, the Max Flow Problem (Wolsey, 1998)) can be solved with a polynomial algorithm, no one to date has claimed to find an efficient (polynomial) algorithm for a general MILP (or BIP) problem.

In the application of subgradient algorithms for solving MILP/ILP problems, the subproblems are generally optimization problems involving integer variables. Since there is no proof or declaration of a polynomial time algorithm for a general MILP/ILP, the subproblems themselves may not be solved in polynomial time. Therefore, no matter what the complexity of coordination, the worst case complexity of subgradient optimization would be NP-hard, when solving a decomposable MILP problem. Therefore, we are not going to emphasize the worst-case behavior of the subgradient optimization algorithm, but rather concentrate on its average-case behavior through the empirical studies in next section.

5.3.2 Empirical Studies

To gain more insight into the scaling behavior and computational efficiency of the proposed subgradient optimization algorithm, four groups of Monte Carlo simulation are performed by solving randomly generated (linear) Binary Integer Programming (BIP) problems[7].

The BIP test problem instance generation scheme is introduced in Appendix A.3.1. The problem instance generation program randomly generates a set of BIP problems with block-angular structure. After some preliminary computational experiments, a reference problem model is determined so that we can observe algorithm performance changes when we vary the problem structural parameters with respect to the reference model. The following set of parameters are used for the reference problem model:

$$\{p = 17, m_0 = 10, m_i = 20, n_i = 15 \quad i = 1, 2, ..., p\}$$

---

[7] Since all the integer programming problems can be equivalently converted to binary integer programming problems (Rao, 1998), the simulation study focuses on BIP problems.

It should be noted that the reference problem is a "well-balanced" decomposable problem with $RSR = 1$. We are taking the same structural analysis approach as discussed in Chapter 2 and 3 with the assumption that each subproblem has been assigned to a distributed CPU.

Again, the equivalent computational time $\mathbf{t}_{eqv}$ for the subgradient optimization algorithm is estimated by summing up the time for solving the coordination problem and the most time-consuming subproblem, assuming a distributed computational environment. As a reference, the generated BIP problems are also solved by a typical centralized MIP solver. Because the computational studies are only designed to investigate the scaling behavior of the proposed subgradient optimization algorithm without any primal solution recovery efforts attempted, the subgradient optimization procedure is terminated when the solution to the LD problem is close to the LD optimal solution. The algorithm is terminated when there is no improvement (e.g., less than 10^{-6}) made in the objective function values (i.e., $||L(\boldsymbol{\lambda}_k) - L(\boldsymbol{\lambda}_{k-1})||$) or in the subgradient vector norms (i.e., $||\hat{\mathbf{s}}^k - \hat{\mathbf{s}}^{k-1}||$) for a number of consecutive iterations [8]. The algorithm will also be terminated when it exceeds the prespecified maximum iteration number, which is usually chosen as an increasing function of the decision variables (e.g., $maxIter = 50 + 20 \times \lfloor \frac{N}{50} \rfloor$, where N is the number of decision variables). It should be noted that, the centralized MIP solver is used to solve BIP primal problems, while the subgradient algorithm is used to solve the LD of the BIP problems. Usually, there is a duality gap between the two solutions. In both cases, ILOG® CPLEX 9.0 MIP solver is used to solve all integer programming problems. In the Monte Carlo simulation for each scenario, the number of problem instances is $200 \times 5 = 1000$, i.e., for each scenario, five runs of simulation are performed and each run solves 200 problem instances generated randomly.

Scenario 1: We fix p and $|I_i|$, change m_0 (see Appendix A.3.2). In this case, we can study the performance of decomposition and coordination with respect to the dimension of linking constraints in equation (5.32).

Figure 5.6 shows that the CCN is relatively insensitive to the dimension of linking

---

[8] Usually, this number must be increased with problem size. In this work, we choose 3 to 5 in our implementation.

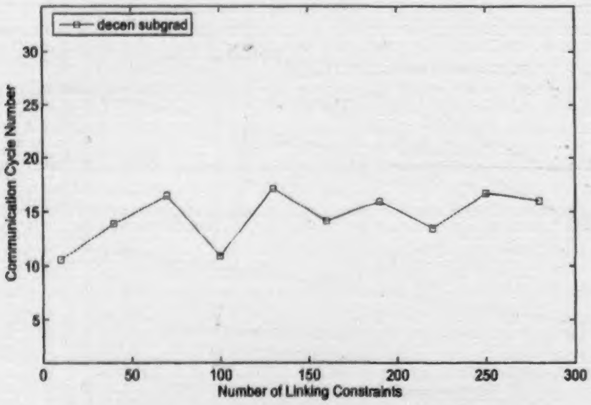


Figure 5.6: Subgradient optimization: CCN vs. number of linking constraints

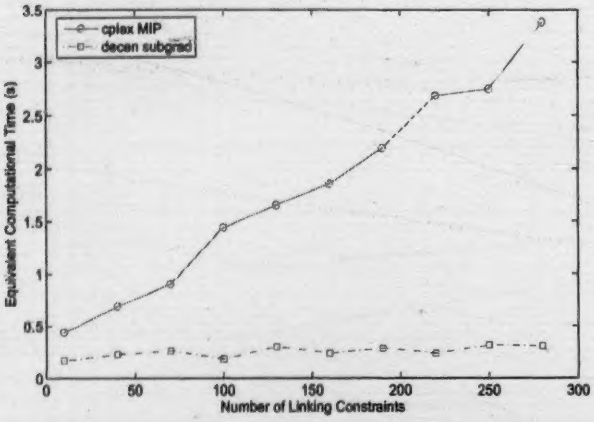


Figure 5.7: Subgradient optimization: computational performance vs. number of linking constraints

constraints, although there is a slight increase of CCN. This is consistent with the average convergence behavior of subgradient optimization methods. It has been observed that the subgradient optimization approaches the optimum fast at the beginning and slows down when it is close to the optimum (Fumero, 2001; Baker and Sheasby, 1999; Bazaraa and Sherali, 1981). Thus, the simulation results imply that the subgradient optimization algorithm takes almost the same number of iterations to reach the desired vicinity of the optimum when the number of linking constraints increases. Note that, as is shown in Figure 5.7, the subgradient algorithm presents better scaling behavior than the centralized MIP solver does.

Scenario 2: For fixed p and m_0, we change subproblem size $|I_i|$ by simultaneously changing m_i and n_i (see Appendix A.3.2). In this scenario, we study the algorithm performance with respect to subproblem sizes.

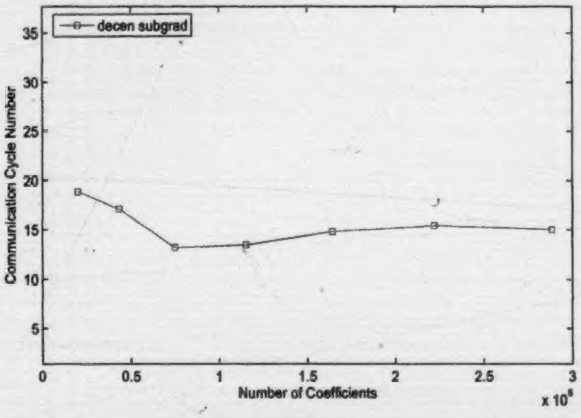


Figure 5.8: Subgradient optimization: CCN vs. subproblem size

The simulation results in Figure 5.8 are again consistent with the known convergence behavior of subgradient optimization. The CCN does not have much dependence on the subproblem sizes; however, since the solution of a larger subproblem is more time consuming, Figure 5.9 shows an increase in the computational time of subgradient optimization algorithm, but its scaling behavior is still better than the centralized MIP solver. It should be noted that the cross-over in Figure 5.9 shows the subgradient algorithm

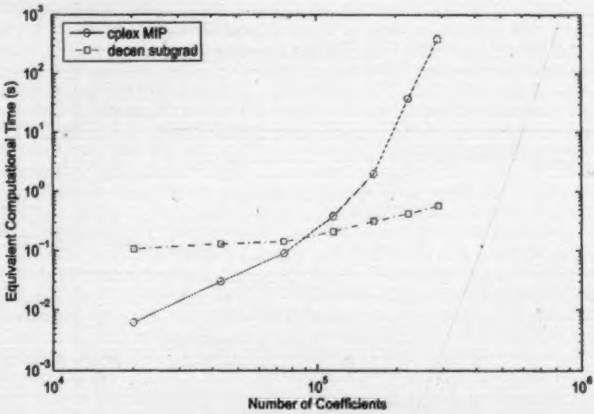


Figure 5.9: Subgradient optimization: computational performance vs. subproblem size

requires less (equivalent) computational time when the subproblem size becomes large (i.e., when the number of coefficients is larger than 10^5).

Scenario 3: We keep m_0, m_i and n_i constant, and change the number of subproblems p (see Appendix A.3.2). In this case, we investigate the performance of the coordination algorithm when more and more subproblems are integrated into the coordination system, assuming a rather well-balanced subproblem computational load.

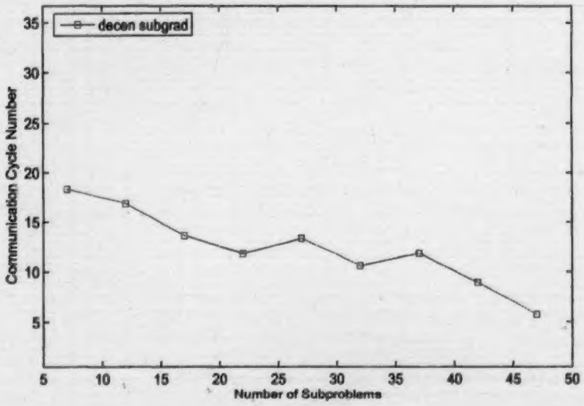


Figure 5.10: Subgradient optimization: CCN vs. number of subproblems

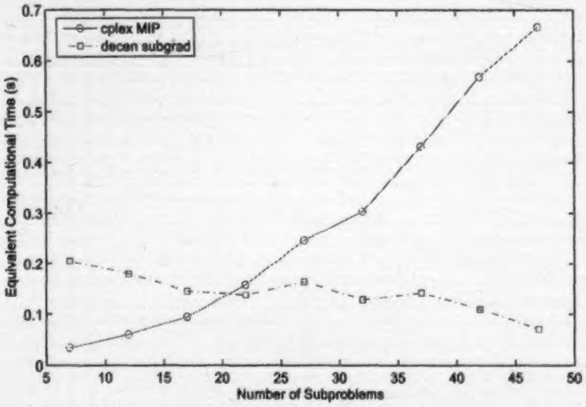


Figure 5.11: Subgradient optimization: computational performance vs. number of subproblems

Figure 5.10 shows that the CCN would decrease when the number of subproblems increases. It should be noted that, when the number of subproblems increases, the number of linking constraints remains the same; thus the impact of linking constraints on the CCN and computational time becomes less significant when the number of subproblems increases. An extreme case is when we have an infinite number of subproblems (*i.e.*, $p \rightarrow \infty$) the impact of linking constraints becomes negligible, thus it becomes a fully decomposable problem and no communication cycle is needed (*i.e.*, $CCN \rightarrow 0$). In other words, the incorporation of a similar-size subsystem does not deteriorate the computational performance, which also indicates good scaling behavior of the subgradient optimization algorithm. Figure 5.11 also shows better computational performance of the subgradient optimization, relative to the centralized MIP, for large numbers of subproblems (*i.e.*, when $p \geq 21$).

Scenario 4: If we fix m_0, $\sum_{i=1}^{p} m_i$ and $\sum_{i=1}^{p} n_i$, i.e., we fix the overall problem size, we can study the influence of relative subproblem ratio (RSR). In this case, we change p by combining subproblems into groups (see Appendix A.3.2) according to different partition patterns of the original BIP problem as we did for LP and QP in Chapter 2 and 3, respectively.

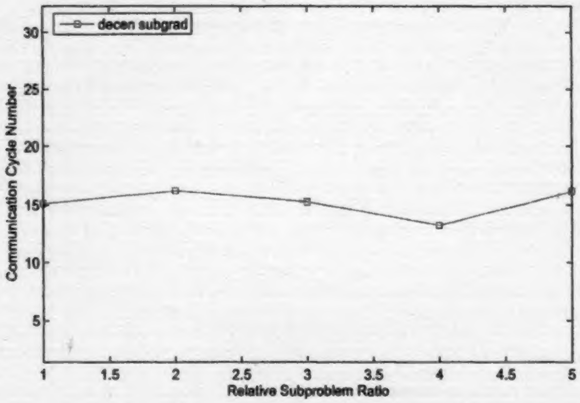


Figure 5.12: Subgradient optimization: CCN vs. RSR

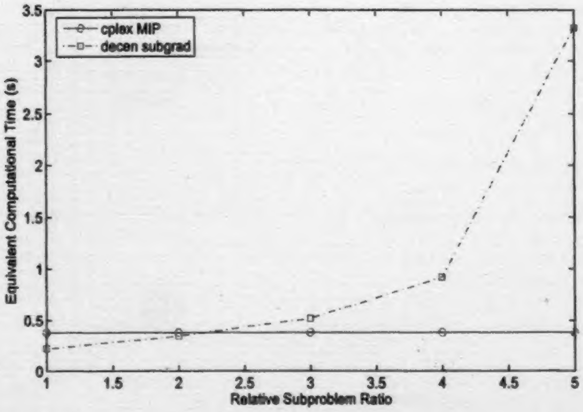


Figure 5.13: Subgradient optimization: computational performance vs. RSR

In Figure 5.12, the CCN shows little dependence of the RSR; while the increase in computational time in Figure 5.13 implies that the increase is mainly due to the increase in solving subproblems (i.e., mainly due to the imbalanced dominant subproblem). We can see that even for the same BIP problem, different decomposition patterns lead to different computational performance although the CCN is not affected very much.

Remarks: The main task of coordination is to efficiently update the multiplier $\boldsymbol{\lambda}$. As heuristic-based strategies are used to calculate the step size and adjust the search direction, the update of $\boldsymbol{\lambda}$ depends more on the heuristics than the solution of subproblems. In this case, the search direction adjustment and step size calculation do not show dependence on the problem structure parameters, which results in a coordination that is insensitive to problem structure. Consistently, from the empirical study, it can be observed that the computational complexity (CCN) of the subgradient optimization algorithm does not have significant dependence on the chosen structure parameters; however, the computational performance is significantly affected by the complexity of subproblems. Therefore, we may conclude that the subgradient optimization algorithm has good scaling behavior and can be used to efficiently solve the LD problem of an MILP or BIP problem.

5.4 Approaches to Primal Solution Recovery

In this section, the development of primal solution recovery heuristic is inspired by the *Interior Path Methods* (Hillier, 1969; Faaland and Hillier, 1979). Although the *Interior Path Methods* were not developed for retrieving primal solution from the solution of the LD problem, it provides the bases for the idea of the proposed *Interior Path Search* heuristic.

The heuristic algorithms developed in Hillier (1969) and Faaland and Hillier (1979) deal with general integer linear programming problems. The algorithms search for good integer solutions in the neighborhood of the optimal solution for the corresponding linear relaxation of the original problem. In some sense, the algorithms work in two phases, where "interior paths" from the optimum of the relaxed problem to some interior points are constructed in Phase 1 and the search for a nearby feasible (integer) solution along the interior paths is performed in Phase 2.

Figure 5.14 shows an illustrative linear integer programming problem, in which the black dots represent the integer solutions. The heuristics aim to generate "good" paths leading from an optimal solution for the corresponding linear programming problem (i.e., an LP relaxation of the MILP problem) into the interior (e.g., point 1 or 2 in the figure) of the feasible region for this problem.

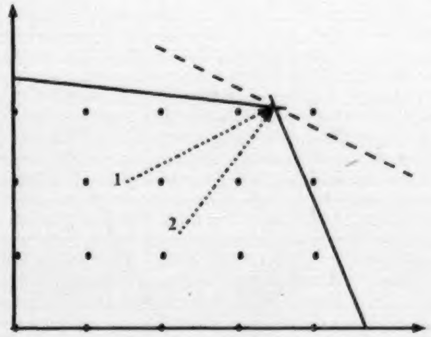


Figure 5.14: Basic idea of interior path methods

Following the generated paths for given step sizes, the integer solutions close to the paths in the feasible region are evaluated. In the end, the best solution recorded in the evaluation procedure is reported as the final solution (optimal or suboptimal) to the original MILP problem.

5.4.1 Primal Solution Recovery Heuristics

This subsection discusses an Interior Path Search method by taking advantage of the solution from subgradient optimization, which is a way to make use of the solution of the LD problem.

For a general mixed-integer programming problem described in equation (5.1), the solution to its Lagrangian dual problem, which is given in (5.3), provides a lower bound on the optimal solution to the original problem for any $\boldsymbol{\lambda} \geq \mathbf{0}$. This lower bound is at least as good as the bound from the solution of the linear relaxation problem (i.e., ignore the integral constraints and treat all the integer variables as continuous variables) of the original problem. The proof can be found in many classical textbooks, such as Beasley (1993) and Wolsey (1998).

For the purpose of illustration, let us consider a two dimensional case. In Figure 5.15, the upper dashed line represents the objective function of the relaxed problem, while the lower dashed line represents the objective function of the LD problem. In this case, the solution to the LD problem provides a better lower bound for the original integer optimization problem. Assuming that we can determine the intersection points 1 and 2, searching along the path (line segment) between point 1 and 2, it is quite possible to find some feasible integer solutions in the vicinity of the line, which is the optimal or a suboptimal solution.

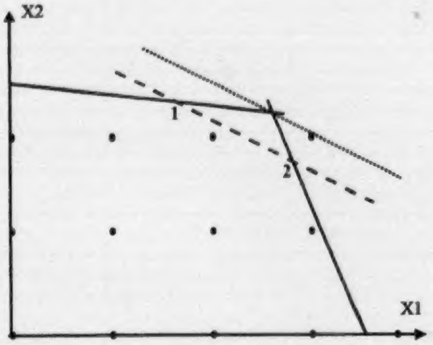


Figure 5.15: Interior path using subgradient lower bound

In particular, for an MILP problem of higher dimension, the (relaxed) feasible region is a polyhedron and the optimal objective function of the LD problem can be expressed by a hyperplane. Therefore, the line segment in Figure 5.15 will be a cross section (a polygon) generated by cutting the polyhedron with the hyperplane. Then we may have many points as the corner points of the polygon. Assuming we can determine all or some of the corner points[9], it is possible to construct lines (interior paths) on the polygon by using some (linear) combination of the corner points. Since the cross section is usually close to the optimal solution or suboptimal solutions, which are in the neighborhood of the optimum, searching along the generated interior paths, it is quite possible to find a feasible suboptimal solution or the optimum.

---

[9] In theory, we can obtain all the corner points of the polygon, but in practice, we may be allowed to

To generate the interior paths in the above scheme, one question to be answered is the generation of end points for a line segment or corner points for a polygon. The following example is used to illustrate one approach to generating desirable corner points. Consider a general MILP problem:

$$\begin{aligned} \text{(P1):}\quad & \min\ \mathbf{c}_1\mathbf{x}_1 + \mathbf{c}_2\mathbf{x}_2 \\ \text{subject to:}\quad & \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 \le \mathbf{0} \\ & \mathbf{x}_i \in \mathcal{X}_i \end{aligned} \tag{5.33}$$

where $\mathcal{X}_i$ contains integer sets. Assumping the lower bound from the solution to the LD problem of P1 is f_{lb}. To make use of the lower bound, we can add one more constraint to the original constraint set and get an LP relaxation of the original problem:

$$\begin{aligned} \text{(P2):}\quad & \min\ \mathbf{c}_1\mathbf{x}_1 + \mathbf{c}_2\mathbf{x}_2 \\ \text{subject to:}\quad & \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 \le \mathbf{0} \\ & \mathbf{c}_1\mathbf{x}_1 + \mathbf{c}_2\mathbf{x}_2 \ge f_{lb} \\ & \mathbf{x}_i \ge \mathbf{0} \end{aligned} \tag{5.34}$$

It may be noted that the problem (P2) is a degenerate problem. Although most commercial optimization solvers can handle this kind of degeneracy, we may modify the problem by introducting perturbations to the elements of the coefficient vector $\mathbf{c}_1$ or $\mathbf{c}_2$[10].

$$\begin{aligned} \text{(P3):}\quad & \min\ \mathbf{c}_1\mathbf{x}_1 + \mathbf{c}_2\mathbf{x}_2 \\ \text{subject to:}\quad & \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 \le \mathbf{0} \\ & \bar{\mathbf{c}}_1\mathbf{x}_1 + \bar{\mathbf{c}}_2\mathbf{x}_2 \ge f_{lb} \\ & \mathbf{x}_i \ge \mathbf{0} \end{aligned} \tag{5.35}$$

---

determine some of the corner points due to computational requirements.

[10] This actually leads to a slight change in the slope of the line segment in Figure 5.15

The resulting problem (P3) can be solved to give a feasible solution (i.e., a point on the boundary) that is close to one of the end points or corner points. By randomly perturbing the coefficients of different decision variables in the augmented constraint (5.4.1), we may end up with a set of points that are supposed to be in the vicinity of the desired corner points. It should be noted that, as is shown in Figure 5.15, we cannot use two points close to point 1 to generate an interior path but one point close to point 1 and one point close to point 2. By recording the indices of active constraints when we solve the problem (P3), we can select a set of desired corner points, which correspond to different sets of active constraints, to construct such interior paths.

For one generated interior path, denoted as path $\mathbf{x}^1\mathbf{x}^2$, where $\mathbf{x}^1$ and $\mathbf{x}^2$ are the end points of the interior path, we can search along the interior path by evaluating the point:

$$\mathbf{x}^j = \lfloor (1-\alpha)\mathbf{x}^1 + \alpha\mathbf{x}^2 \rfloor, \quad \text{where } 0 \leq \alpha \leq 1 \tag{5.36}$$

where "$\lfloor\rfloor$" is the operation for rounding all integer variables to obtain an integer solution. The step size α can be increased at a fixed increment or according to some other update schemes. Then feasibility test will be performed for each evaluated point $\mathbf{x}^j$, i.e., we only record the set of solutions that satisfy all constraints in problem (P1). It should be noted that the integral constraints have been satisfied through the operation in (5.36). By searching along all the generated interior paths, the best integer solution $\mathbf{x}^*$ is recorded and reported as the final solution.

5.4.2 A Decentralized Optimization Framework

So far we have obtained an improved subgradient optimization algorithm as well as a primal solution recovery heuristic, which contribute to the two essential parts of a subgradient optimization technique for solving MILP problems. As is discussed in previous sections, on a distributed computing platform, the distributed version of the improved subgradient algorithm can be used to efficiently solve the LD of a block-structured BIP (or MILP) problem. In addition, to fully utilize the distributed computing environment, the Dantzig-Wolfe decomposition algorithm can be used to solve decomposable LP problems (P3) for generating the corner points of desired interior search paths. Next, a decentralized optimization framework is proposed for implementing the MILP decomposition algorithm.

Based on our experience with Dantzig-Wolfe decomposition (Cheng *et al.*, 2004) and Price-driven coordination methods (Cheng *et al.*, 2005*b*), the proposed framework of decentralized optimization with coordination shares the same information exchange pattern as is shown in Figure 5.16.

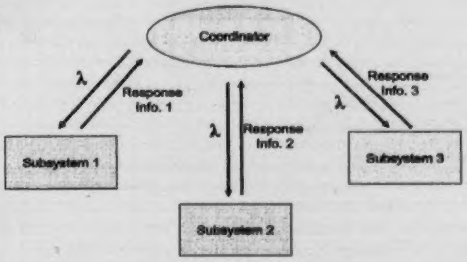


Figure 5.16: Diagram of a generalized coordination framework

The information, which flows between the coordinator and each subsystem, includes multiplier information from the coordinator to subsystems and response information from each subsystem to the coordinator. The multiplier $\boldsymbol{\lambda}$ usually contains sensitivity information with respect to linking constraints, while the response information contains the influence of the multiplier changes on subproblems' solutions in some way (implicitly or explicitly). Note that, there exist a set of optimum multipliers $\boldsymbol{\lambda}^*$ that solves the overall problem. A major difference is the way the coordinator updates the "price" multiplier $\boldsymbol{\lambda}$ as the coordinated system converges to the optimal solution. In the Dantzig-Wolfe decomposition, a linear program (RMP) is solved by the coordinator to update the "price" information; in the price-driven coordination method, a system of equations is solved by the coordinator for updating the multiplier; while dealing with MILP problems, Lagrangian relaxation (Lasdon, 2002; Beasley, 1993) and subgradient optimization methods (Fumero, 2001; Camerini *et al.*, 1975) are used to update the "price".

Shown in Figure 5.17, the proposed decentralized optimization framework for solving MILP problems consists of three phases:

1. Subgradient optimization (solution of LD problem): subgradient multipliers are updated by the coordinator while the subproblems are solved on local computing

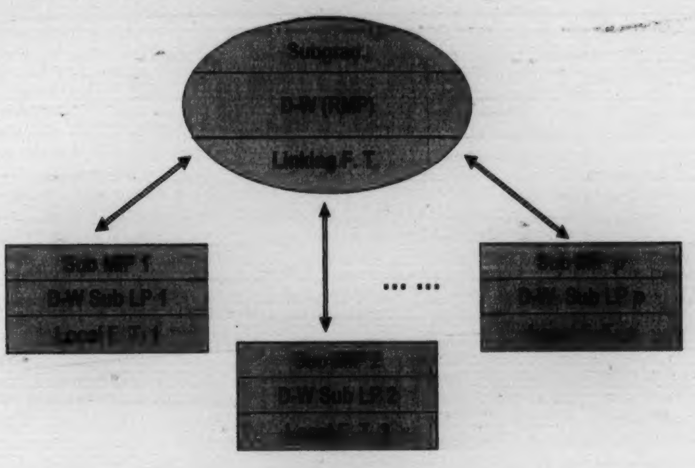


Figure 5.17: Decentralized framework for MIP

platforms;

2. Interior path generation via Dantzig-Wolfe decomposition algorithm: the solution from the first phase is used to construct a new LP problem, and then Dantzig-Wolfe decomposition algorithm is implemented in this decentralized computing environment to obtain the corner points for generating the interior paths;

3. Interior path searching and feasibility test: the coordinator is responsible for searching along the interior paths and finding the candidate solutions, then the feasibility test for linking constraints is performed by the coordinator while the feasibility test for local constraints is performed by each local subproblem.

The second and third phases are focused on retrieving the primal solution. Usually, for computation and communication consideration, only after the feasibility test for linking constraints is passed will the local feasibility test be performed.

5.5 Case Study: Truck Allocation Problem

This subsection gives an illustration of the application of the proposed decentralized optimization framework (including Lagrangian relaxation, subgradient optimization and primal solution recovery methods) in solving an industrial MIP problem.

5.5.1 Application Problem

Canada's oil sands[11] contain the biggest known reserve of oil in the world. Since the 1920's, open pit mining has been central to oil sands development. Nowadays, the oil sands industry employs the truck-and-shovel technology predominantly in its open-pit mining operations, as is shown in Figure 5.18. It has been widely recognized that the operation of the huge trucks and shovels contributes significantly to the overall mining operation cost (Ta, 2002; Ta *et al.*, 2005). In order to reduce the cost of mining operations, it is desirable to take any opportunity to reduce the cost of the truck and shovel operation.

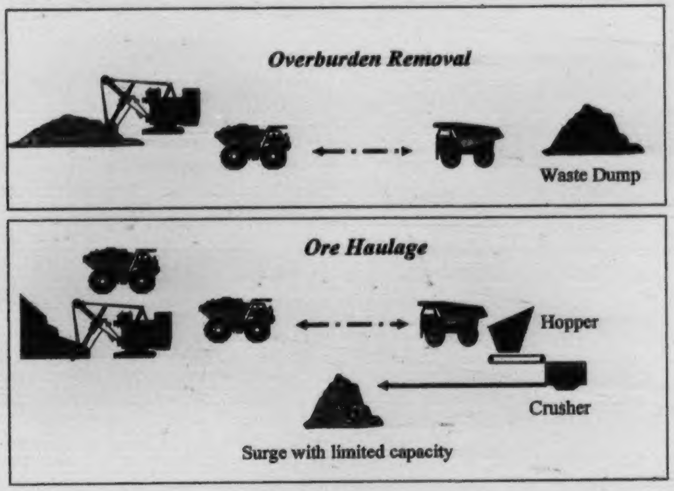


Figure 5.18: Truck-and-shovel oil sands mining operations

In the oil sands industry, a high level performance of mining operations is ensured by an effective deployment of available trucking resources to maintain a steady, reliable supply of ore and a timely removal of overburden. In practice, many companies develop and implement integrated multi-stage decision support systems to help achieve optimal operation of trucks. The multi-stage decision making usually involves two stages: an allocation of truck resources for given production requirements, and a real-time dispatch of the allocated trucks to specific routes. By treating the problem as a whole, a common approach is to solve the allocation problem via mathematical programming techniques and the dispatching problem via heuristics (Elbrond and Soumis, 1987; Lizotte *et al.*, 1989; Xi and Yegulalp, 1993; Temeng *et al.*, 1997); however, in Ta *et al.* (2005), it was recognized that the allocation and real-time dispatch are separate tasks. With the increasing size of mining operations, it is more practical to treat the above two problems separately. In this chapter, our work concentrates on truck allocation phase and ignores the dispatch task.

---

[11] Oil sands are mixtures of sand, water, clay and crude bitumen.

In today's oil sands operations, truck and shovel mining presents interesting challenges for operations optimization, including: multiple, competing and/or conflicting objectives, such as minimizing the operation cost (e.g., the usage of raw materials and energy) and maximizing production, among operating units and subsystems; limited availability of resources to be effectively allocated to competing subsystems; large sets of constraints defining the feasible domain; a requirement for smooth transition between consecutive optimization executions; and uncertainty in the system parameters. For complex mining operations, the truck allocation problem usually has a large scale and decomposable structure. A standard approach to problems of this type would decompose the large-scale optimization task of assigning truck resources to various activities. This fully decentralized approach would treat the optimization of ore production, waste removal and equipment maintenance, independently. Although this approach is commonly implemented with customized modifications, it cannot guarantee that the resulting operation is optimal with respect to the entire mining operations. It has been recognized that it is necessary and beneficial to have coordination between ore hauling and overburden removal processes to achieve mine-wide optimal operations.

In this section, a coordination approach is proposed to guarantee the coordinated, decentralized optimization system will find the optimal operation or suboptimal solution, which is close to the optimal operation.

5.5.2 Decentralized Optimization with Coordination

This section discusses the viability of a decentralized decision support system for existing mining practice. An optimization-based decision support framework is developed by taking advantage of decentralized optimization techniques.

Decentralized Decision Support

Current oil sands mining can be classified into three functional processes: ore hauling, overburden removal, and mechanical maintenance. These individual operations can take place over a wide area, which may lead to a decentralized real-time decision making procedure. The decentralized optimization framework has many advantages. It naturally takes advantage of the special organizational structure and can provide an elegant economic explanation for decentralized management. Via decentralization of decision making, a large-scale problem is decomposed into smaller and easier problems, and distributed computing environment can be fully utilized.

The existing decentralized optimization for mining operations, however, may not be able to provide enterprise-wide optimal operations, because it fails to consider the interactions between the multiple processes, e.g., the share of truck and shovel resources. Obviously, non-optimal or even conflicting decisions could be made through this decentralized optimization. In current practice, priorities are usually pre-specified for each process to avoid the conflicts in decision-making.

Coordination for Decentralized Optimization

To make the individual optimizers work more efficiently, it is desired to establish coordination mechanism to deal with the interactions between operating processes. It is possible to implement a centralized decision support system, in which an optimization problem is formulated and solved for the entire mining operations, including inter-unit interactions. This approach yields the optimal operations; however, it requires centralized database management and computing environment, which centralizes the risk of decision-support system failure. This approach results in an "all-or-nothing" system, which has been identified as being impractical for large-scale operations. Alternatively, to retain the desirable features of the decentralized optimization, a coordinator can be added to the original system to handle the interactions and achieve the optimal operations.

Challenges and Opportunities

The main challenge lies in solution of the coordination problem. It has been proposed that a good model for mining operations should be a hybrid system involving both continuous and discrete variables. Thus, the coordination problem is a mixed-integer programming problem. It is well known that a general MIP problem is very difficult to solve, due to the combinatorial nature of problems involving integer variables[12]. An MIP solver which employs a centralized optimization scheme may encounter serious computational problems. For example, as one of the best known commercial optimization software, the CPLEX 9.X solver for MIP problems[13] may fail to prove integer optimality and run out of memory even for a problem containing only 30 binary variables. This occurs when the branch-and-bound algorithm produces a tree having over one billion nodes and the termination criteria have been set improperly (GAMS, 2004). The situation becomes worse with a more complex model and more constraints. In this case, the decentralized optimization framework can be considered when: 1) it is expensive to implement a centralized computing environment (e.g., a high performance computer with huge memory) that can guarantee the computational efficiency and reliability requirements; or 2) the problem has separable structure and cheap distributed computing environment (e.g., a network of PCs) is available. Furthermore, a distributed optimization approach can provide a more reliable and extendable optimization and control system.

Thus, a major challenge is the development of an effective coordination mechanism for MILP decomposition and coordination strategies, i.e., an efficient way of driving the solutions of the subsystems to the overall optimum. A good coordination mechanism is a decisive factor in ensuring the feasibility and applicability of the resulting coordination system. Due to the discrete nature, no sensitivity information is available for use in the coordination mechanism. All these issues are tackled by the proposed decentralized optimization framework.

---

[12] Any integer variable can be represented by a combination of "0" and "1" binary variables.

[13] CPLEX is a product of ILOG®, which implements branch-and-cut search algorithms that includes the latest research on cut and presolve techniques.

5.5.3 Illustrative Case Study

For the purposes of illustration, our focus is on the coordination of ore hauling process and overburden removal process. Therefore, the following subsections give a simplified mathematical description of the processes of interest[14].

Truck Allocation Problem Formulation

Assuming that the truck allocation problem is to efficiently distribute the available trucks to the individual processes: ore hauling and overburden removal. To show the problem's special structure, a centralized formulation is firstly given in equations (5.37) to (5.41). Note that, for simplicity, the following formulation is based on deterministic optimization and considers one shift operation. A more advanced formulation for handling process uncertainties can be found in Ta *et al.* (2005). By defining the decision variables $\mathbf{x}$ as the set[15] of trucks allocated for ore haulage and $\mathbf{y}$ as the set of trucks allocated for overburden removal, the objective is to minimize the operating cost of trucks while satisfying the ore demand from extraction process and overburden removal requirements.

---

[14] While the models may not reflect all the complexity of the actual mining operations, they are representative and the simplicity allows clear demonstration of the optimization scheme

[15] Including the number of trucks and type of trucks

$$\min \quad \mathbf{c}_h^T\mathbf{x} + \mathbf{c}_r^T\mathbf{y} \tag{5.37}$$

subject to:

$$\mathbf{I}_1\mathbf{x} + \mathbf{I}_2\mathbf{y} \leq \mathbf{r} \tag{5.38}$$

$$\mathbf{B}_1\mathbf{x} \geq \mathbf{d} \tag{5.39}$$

$$\mathbf{B}_2\mathbf{y} \geq \mathbf{w} \tag{5.40}$$

$$\mathbf{x},\ \mathbf{y} \in Z^+ \tag{5.41}$$

where $\mathbf{c}_h$ contains the corresponding operating cost of trucks for ore hauling process for one shift, while $\mathbf{c}_r$ is for overburden removal. Inequality (5.38) represents the restrictions on truck availability for the overall mining operations. Constraints (5.39) and (5.40) are ore production and over burden removal requirements, respectively. This is not the only possible formulation, but it is a reasonable formulation that matches the current truck allocation practice.

The above optimization problem has a block-wise structure. By excluding the linking constraints (5.38) (i.e., the fleet resource), the problem can be decomposed into two independent optimization subproblems, associated with $\mathbf{x}$ and $\mathbf{y}$, respectively. For each process (i.e., ore hauling or overburden removal), without being aware of the workings of the other, each optimizer tries to maintain a minimal operating cost while satisfying the production requirements.

Performance Comparisons

To investigate the necessity and effectiveness of the coordination, a truck allocation problem is solved by three optimization schemes: the centralized, decentralized, and coordinated strategies. For the purpose of comparison, the centralized optimization is used as a benchmark for the optimal operations.

$$\min \quad \sum_{k=1}^{K}\sum_{i=1}^{I} c_h^k x_{ki} + \sum_{k=1}^{K}\sum_{j=1}^{J} c_r^k y_{kj} \tag{5.42}$$

Subject to:

$$
\begin{aligned}
&\sum_{i=1}^{I} x_{ki} + \sum_{j=1}^{J} y_{kj} && \leq \mathbf{r}(k) \quad k = 1,2,3 \\
&\sum_{k=1}^{K}\sum_{i=1}^{I} \frac{60}{T_h^i} L_k x_{ki} && \geq d \\
&\sum_{k=1}^{K} \frac{60}{T_h^i} L_k x_{ki} && \leq Thr(i) \quad i = 1,2,3 \\
&\sum_{k=1}^{K}\sum_{j=1}^{J} \frac{60}{T_r^j} L_k y_{kj} && \geq w \\
&\sum_{k=1}^{K} \frac{60}{T_r^j} L_k y_{kj} && \leq Thr(j) \quad j = 1,2 \\
&x_{ki},\ y_{kj} \in Z^+
\end{aligned}
$$

In the above problem, k represents the truck type (i.e., $K = 3$ says that three types of trucks are in operation), and I and J are the number of shovels for ore haulage and overburden removal, respectively. Thus, x_{ki} represents the number of type k trucks allocated to shovel i for ore hauling, while y_{kj} is the number of type k trucks allocated to shovel j for overburden removal. $\mathbf{r}(k)$ is the availability of type k trucks. c_h^k is the operating cost for running a type k truck for ore hauling per unit time, while c_r^k is the cost for overburden removal per unit time. T_h^i is the cycle time for a truck's[16] roundtrip from an ore shovel i to the corresponding dump pocket; T_r^j has similar meaning in overburden removal process. L_k is the load of type k trucks.

Table 5.1 and Table 5.2 give the data for the truck allocation problem[17], where three shovels are used for ore haulage and two shovels for overburden removal.

Obviously, when the decisions for ore hauling and overburden removal processes are independent and the shared resources are not considered, conflicting decisions may be made. Thus, priorities should be prespecified for each process to gain feasible operations.

---

[16] The same cycle time for different types of trucks is assumed when they are allocated to the same route, i.e., no overtaking is allowed.

[17] The parameter values in the case study do not represent the actual mining operations, and the case study is set up to test the proposed method.

Table 5.1: Parameters for trucking

truck type	load (tonnes)	operational cost ($/hr)	truck fleet size
$k = 1$	240	1000	15
$k = 2$	320	1300	10
$k = 3$	360	1400	8

Table 5.2: Other parameters

shovel no.	throughput (tonnes/hr)	cycle time (min)	others
$i = 1$	4000	25	Ore demand
$i = 2$	5000	35	$d = 12000$ (tonnes/hr)
$i = 3$	4000	30	
$j = 1$	4000	32	OB removal
$j = 2$	3000	25	$w = 6100$ (tones/hr)

Note that, even when priorities are established, infeasible solutions may also occur especially if the common resource constraints are stringent. In this study, as in practice, ore hauling is given a higher priority, i.e., the decision maker ensures ore hauling resources are satisfied first, and overburden removal uses the remaining trucking resources.

Table 5.3: Different optimization strategies for truck allocation problem

	central.	coordinated	decentral.(pri)	decentral.(no pri)
Ore hauling obj.($)	24300	24700	24200	24200
OB removal obj.($)	11900	12100	13000	11600
overall obj.($)	36200	36800	37200	35800 (*Infeasible*)

Table 5.3 reports the solutions of the truck allocation problem resulting from the three different optimization schemes. It should be noted that, when no priorities are pre-specified, the decentralized scheme without considering the linking constraints, yields the lowest operating cost for both ore hauling and overburden removal; however, it provies a solution that leads to infeasible operations, i.e., contradictory decisions occur in the allocation of common resources. With the pre-specification of priorities, the decentralized

scheme makes a little improvement to get a feasible solution, but the total operating cost is higher in order to bring the solution to feasibility. If we look on prioritizing as one means of coordination, it is not the optimal way to conduct coordination. The proposed coordinated, decentralized optimization provides "optimal" coordination of the two interacting subproblems, guarantees a feasible solution, and achieves an operation level close to the real optimum operations (i.e., operations provided by the centralized optimization scheme). To achieve optimal operations through coordination, it is required to have proper modeling of interactions and a well-designed coordination process. It should be noted that, unless you make an exhaustive enumeration, an MILP algorithm cannot guarantee to find the real optimum. Based on current heuristics for the decomposition and coordination algorithm, there is no guarantee of finding the true optimum, but the proposed heuristic-based algorithm yields a solution close to the true optimum.

Remarks: For the purpose of illustration, the truck allocation problem for the case study is not really in large scale. All the optimization schemes can solve the problem in seconds, and thus the computational effort is not emphasized. In addition, this work develops a fairly general and flexible framework for decentralized decision support system, in which many heuristics are incorporated into the decomposition and coordination algorithm for solving the resulting optimization problem. Based on such an optimization framework, better heuristics can be introduced or developed to improve the computational performance to satisfy specified application requirements.

5.5.4 Discussions

Because of the desirable features of decentralized optimization approaches, the optimization problem can be formulated in a module-based fashion such that the resulting coordination system can be easily extended and maintained, and can provide reliable behavior.

The foremost step in the problem formulation is to identify and model the interactions among multiple operating units, i.e., the linking constraints. For a specified purpose, an appropriate subset of those interactions should be identified and modeled, and be used in the coordination of subsystems. For example, the available truck resources may serve the purpose of linking constraints in the truck allocation problem when considering the mining operations only; however, when the plant-wide operation is considered, the balance between the ore demand from Extraction operating units and the ore supply from Mining should be included as a linking constraint. The overall process of Oil Sands mining and extraction operations contains both continuous and discrete processes. There are linking constraint sets associated with integer variables, such as the number and types of trucks and shovels. This unavoidably brings big challenges to the efficient coordination of subsystems.

When the linking constraints are identified and formulated, the major types of information to be exchanged between the coordinator and subsystems is also determined. Then, this information is incorporated into the formulation of subproblems in the form of a parameter (e.g., price vector λ), which is updated by upper level coordination. In this work, we have proposed a multi-level model-based optimization framework for mining operations, in which two-directional (in both time and space) coordination has been proposed. To give a conceptual idea, Figure 5.19 illustrates a two-level coordination framework for Oil Sands operations. In the figure, the "plant-input" can be the production plans established by business and marketing departments, which are used as the targets of process operations (mining and extraction). The "plant-output" information at different levels contains the measurements and data that reflect the state of process operations at

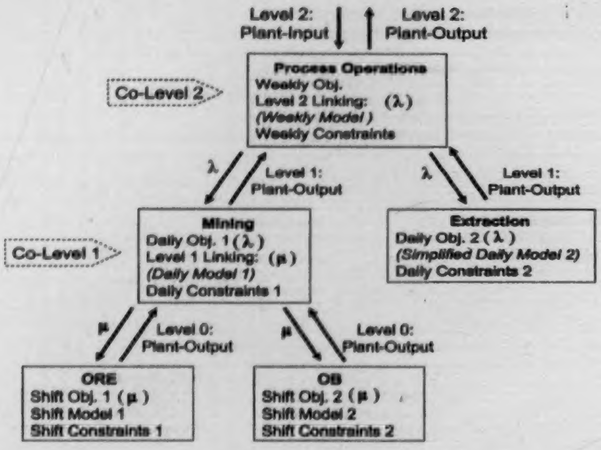

Level 2:
Plant-Input
Level 2:
Plant-Output
Process Operations
Weekly Obj.
Level 2 Linking: (λ)
(Weekly Model)
Weekly Constraints
Co-Level 2
λ
Level 1:
Plant-Output
λ
Level 1:
Plant-Output
Mining
Daily Obj. 1 (λ)
Level 1 Linking: (μ)
(Daily Model 1)
Daily Constraints 1
Co-Level 1
Extraction
Daily Obj. 2 (λ)
(Simplified Daily Model 2)
Daily Constraints 2
μ
Level 0:
Plant-Output
μ
Level 0:
Plant-Output
ORE
Shift Obj. 1 (μ)
Shift Model 1
Shift Constraints 1
OB
Shift Obj. 2 (μ)
Shift Model 2
Shift Constraints 2

Figure 5.19: Model-based coordination framework

a corresponding operating unit. For example, level-0 plant-output is from ore hauling or overburden removal units, and it may contain the information regarding the demand for trucks and ore production level based on that truck demand. As we have discussed previously, the multipliers λ and μ include some sensitivity information with respect to the linking constraints involved with multiple operating units. For example, when the level-1 linking constraints are the available truck resources for mining operations, the multipliers μ can be some artificial prices for the usage of trucks. In the proposed framework, the upper level objective function can be defined as a time integration of lower level objective function. We use brackets in expressing the model (e.g., (daily model)) to show that the models at level k are only used to take part in level $(k + 1)$ coordination. An example is that, when we perform level-1 coordination, the daily model in level-1 coordinator will not be used.

5.6 Chapter Summary

Nowadays, many industrial operations become highly integrated and in large scale, whose high-level performance may be supported by the solution of large-scale optimization problems involving integer variables.

As many large-scale MIP problems have special structure that allows decomposition of the original problem into smaller subproblems, decomposition and coordination strategies for solving large-scale MIP problems are investigated in this chapter. Subgradient methods are widely used in solving decomposable large-scale MIP problems for ease of implementation, but their convergence behavior requires further improvement for some applications. In this work, by incorporating the concept of a feasible search direction, a new heuristic has been developed to adjust the search direction in subgradient optimization methods. The new heuristic provides several favorable properties, which have been proved to improve the convergence behavior of subgradient methods.

By introducing the "best" heuristics in the literature, as well as the proposed direction modification strategy, an enhanced subgradient optimization algorithm is developed and given in form of a centralized version and a distributed version. The structural complexity analysis provides insight into the scaling behavior of the proposed subgradient algorithm, and implies the feasibility of the algorithm in industrial applications.

In addition, a novel *Interior Path Search* heuristic has been developed for retrieving primal solutions from the LD problem solution. By taking advantage of the Dantzig-Wolfe decomposition strategy, the proposed heuristic can make full use of a distributed computing platform. As a result of the enhanced subgradient optimization algorithm and the primal solution recovery heuristic, a decentralized optimization framework is proposed for solving large-scale decomposable MILP problems.

Furthermore, the decentralized optimization framework has been applied to a decentralized decision support system for truck allocation in Oil Sands mining operations. Within the decentralized optimization framework, the enhanced subgradient algorithm and Interior Path Search heuristic can efficiently solve the associated MILP problem. The case study results show that the proposed optimization framework can be a viable technique for solving industrial MILP problems.

"Learning without thought is labor lost; thought without learning is perilous."

– by Confucius

6

Conclusions and Future Work

6.1 Summary and Conclusions

In the past few decades, large-scale optimization has been identified as one of the most efficient ways to ensure high-level operational performance. Optimization can make an organization more competitive by maximizing production profit and minimizing cost. With growing understanding of the chemical and physical rules underlying each process, it is possible to describe a system using more detailed, high fidelity models. Optimization problems based on these more informative but more complex models can result in fairly high dimensional optimization problems.

For solving large-scale optimization problems, the decomposition of large-scale (complex) systems (or problems) into a number of interconnected subsystems (or subproblems) has been recognized to be beneficial. In this thesis, partitioning of the system is used to achieve not only a reduction of numerical complexities of problem but also conceptual simplification in understanding a large-scale system. Besides the purely computational difficulties arising in large-scale systems, there is an equally important problem of understanding the effects of system (or problem) structure on the performance of decomposition algorithms.

This work is intended to make a contribution to a wide range of application areas, including optimization and process control. In particular, the work has been focused on the study of three classes of large-scale optimization problems: linear programming (LP), quadratic programming (QP), and mixed-integer linear programming (MILP) problems.

Large-scale optimization problems frequently occur in industrial applications, and many of them naturally present special structure or can be transformed to such a special structure. To take advantage of a distributed / decentralized computing environment, some decomposition and coordination methods have the potential to solve specially structured, large-scale optimization problems at a reasonable speed. This work identifies the best available decomposition strategies through literature review and computational study, as well as develops more efficient decomposition algorithms whenever needed. Using an empirical complexity analysis approach, the scaling behavior and computational efficiency of the decomposition strategies are investigated. The complexity study provides guidelines to the practical applications of the decomposition and coordination methods.

Chapter 2 investigates decomposition strategies for large-scale LP problems. In particular, Dantzig-Wolfe decomposition strategies based on single-column and multi-column generation techniques are studied. New insight into the relationship between computational performance and problem structural parameters is gained through a comprehensive empirical study of the scaling behavior of Dantzig-Wolfe decomposition algorithms.

In Chapter 3, for solving large-scale QP, an efficient price adjustment scheme is proposed that uses Newton's method to compute the price vectors, which takes advantage of the sensitivity information from subproblem solutions. With the proposed price adjustment scheme, the computational performance of price-driven coordination methods is substantially enhanced when solving large-scale QP problems. A complexity study provides an understanding of the computational behavior of the enhanced price-driven coordination and provides guidelines for the implementation of the algorithms in industrial practice.

Chapter 4 demonstrates novel applications of the decomposition and coordination strategies discussed in Chapter 2 and Chapter 3. For industrial plant-wide control, this work provides new formulations of plant-wide MPC target calculation to achieve plant-wide optimum operations. The proposed coordinated, decentralized MPC framework consists of individual MPC subsystems and a coordinator. The Dantzig-Wolfe decomposition and price-driven coordination methods are applied to the design of coordination system, for LP-based and QP-based MPC target calculation, respectively. By modeling the linking constraints with the off-diagonal element abstraction method or interstream consistency method, the resulting optimization problems fall into a category of separable LP or QP problems with linking constraints, which can be efficiently solved by the decomposition and coordination methods.

Chapter 5 extends our study to large-scale MIP and focuses on MILP problems. By incorporating the concept of a feasible search direction, new heuristics are proposed to modify the search direction in subgradient optimization methods. With the proposed direction modification strategy, an improved subgradient optimization algorithm is developed, which can provide faster convergence. The complexity study provides insight into the scaling behavior of the proposed subgradient optimization algorithm, and illustrates the utility of the algorithm for industrial applications. Furthermore, primal solution recovery techniques are also investigated. To accommodate a distributed computing environment, an Interior Path Search method has been proposed for primal solution recovery by taking advantage of the Dantzig-Wolfe decomposition strategy. By applying the coordination mechanism in subgradient optimization and the primal solution recovery heuristics, a multi-level optimization framework is developed for the decentralized decision support of mining operations, which may involve coordination problems that can be formulated as large-scale MILP problems.

In general, decomposition strategies show high computational efficiency whenever a distributed computing power is available. Based on the coordination techniques and algorithms developed in this work, to improve the computational performance of the decomposition strategies in order to meet specific computational requirements, a decentralized optimization framework has been proposed to embed the decomposition strategies in a distributed computing environment. In Chapter 4 and Chapter 5, typical process control and mining operations optimization problems are solved to illustrate the implementation of the decentralized optimization framework. These case studies show the viability of the proposed coordinated, decentralized optimization techniques.

6.2 Suggestions for Future Work

A number of challenges remain in both the development and application of the decomposition strategies for large-scale optimization.

Coordination in Plant-wide Hybrid Control Systems

In much of this thesis, we have focused on the control and optimization of processes with continuous variables; however, in many industrial operations, such as control and scheduling in power plants (Gallestey *et al.*, 2003), chemical processes including logic switch control (Stursberg and Panek, 2002), actions like switch on/off, mode transitions, products selections, and raw materials selection, *etc.*, are modeled as discrete variables, so it is natural to formulate a constrained MPC with discrete variables.

For modeling and control, a *Mixed Logical Dynamical* (MLD) model is synthesized to describe the processes by Bemporad and Morari (1999). They proposed a framework for controller design when the process includes both dynamical and logical variables and are subject to operating constraints. Usually, continuous dynamic models are obtained from algebraic and differential equations based on our understanding of process chemistry and physics. On the other hand, the existence of logical components introduces integer variables to the problem formulation. So it is quite natural to formulate an MPC problem based on mixed-integer quadratic programming (MIQP). Moreover, Stursberg and Engell (2002) proposed a hybrid automaton which models autonomous switching between different nonlinear dynamics and includes both discrete and continuous control inputs.

Using linearization techniques, we can transform the dynamics into linear discrete time models. Then the task of generating an optimal control law to drive the system to a desired target region can be formulated as an MIP problem.

With the increasing maturity of hybrid control techniques, MPC technology has extended its application to processes involving discrete input and output variables. In plant-wide control and optimization, the coordination of decentralized MPC has been recognized as a beneficial approach to improve plant-wide control performance. Similarly, when decentralized hybrid MPC is employed, the coordination of such MPC controllers becomes a challenging but promising opportunity to improve control performance. Interactions between subunits, which may then include discrete variables, can be abstracted or identified via advanced process control and system identification techniques. Naturally, the overall problem can be formulated as a large-scale MILP or MIQP problem with block angular structure. A coordination system can be designed for decentralized hybrid control systems by employing decomposition strategies for large-scale MIP.

Design of Hybrid MPC Systems Coordination

By applying a coordination mechanism, a framework for coordination system design can be developed for two-level hybrid MPC systems that include a steady-state target calculation and a dynamic control calculation. For industrial applications, it is important to have a good understanding of the coordination complexity, scalability behavior of the coordination algorithms and coordinated system stability. Industrial testbeds can be easily found in power systems, pharmaceutical plants where logic control systems are often used, or some polymer plants which have grade transition or mode selection operations.

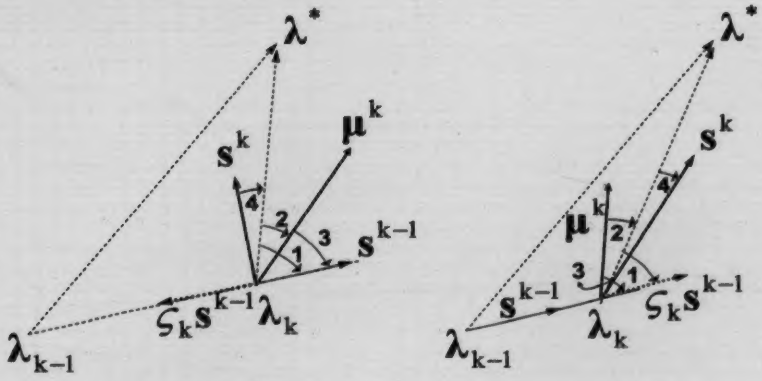
Different from the interactions in continuous processes or models, interactions in a hybrid system can be represented by pure integer constraints, pure continuous variable constraints, and hybrid constraints. Quite often, we have the ability to select appropriate interactions for coordination. To design a coordination system for decentralized hybrid MPC systems, it is desired to gain insight into the influence of coordinating different kinds of interactions. Given a hybrid process with a decentralized control system, before designing the coordination system, it is better to understand what will happen to the complexity of coordination if one additional constraint with both continuous and discrete variables is added. It is necessary to understand the performance loss when such a contraint is ignored, by comparing different coordination system designs. If the ignorance of an extra constraint does not cause too much performance loss, or the inclusion of an extra constraint increases the complexity of coordination too much, it may be considered better not to add such a constraint; however, when such a constraint has strong impact on the control performance (i.e., it cannot be ignored), we may need to determine how to relax the discrete variables and compensate for its integrality.

Furthermore, it should be noted that, if a moving-horizon is considered, the problem dimension of control calculations will be much higher than that in target calculations, thus different information flows should be determined in these two cases. One question may be whether all the control move information should be passed to the coordinator or just part of them. It can be a trade-off between control performance and computational efficiency throughout the coordination, and it may cause big difference in the computational load for coordination and communication load on industry networks.

Further Adjustment for Subgradient Direction

In the search direction adjustment scheme given in formula (5.9), the situation where $\mathbf{s}_{k-1} \cdot \boldsymbol{\mu}_k < 0$ was considered; however, this scheme did not make any change to the direction when $\mathbf{s}_{k-1} \cdot \boldsymbol{\mu}_k \geq 0$, which implies that the current gradient direction $\boldsymbol{\mu}_k$ is a good direction. But this may not be always true as is shown in Figure 6.1. Recall the following facts of angles in the figure. The angle ($\angle 1$) spanned by vector $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$ and $\mathbf{s}^{k-1}$ is an acute angle; the angle ($\angle 2$) spanned by vector $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$ and $\boldsymbol{\mu}^k$ is an acute angle. It is possible that the angle ($\angle 3$ in Figure 6.1) spanned by vector $\boldsymbol{\mu}^k$ and $\mathbf{s}^{k-1}$ is an acute angle, i.e., $\mathbf{s}_{k-1} \cdot \boldsymbol{\mu}_k \geq 0$. In both cases shown in Figure 6.1, intuitively, we may find a better direction $\mathbf{s}^k$ which is a combination of $\boldsymbol{\mu}^k$ and $\mathbf{s}^{k-1}$, and the resulting angle ($\angle 4$) spanned by $\mathbf{s}^k$ and $\boldsymbol{\lambda}_k\boldsymbol{\lambda}^*$ is a smaller acute angle, i.e., $\angle 4 < \angle 2$.

Based on the geometric interpretation, we can see it is possible to further improve the direction $\boldsymbol{\mu}^k$, even when $\boldsymbol{\mu}^k$ and $\mathbf{s}^{k-1}$ form an acute angle. It may be desired to develop a heuristic that can handle both of the situations shown in Figure 6.1. For example, the


λ^*
$\mathbf{s}^k$
μ^k
4
2
3
1
$\mathbf{s}^{k-1}$
$\varsigma_k \mathbf{s}^{k-1}$
λ_k
λ_{k-1}
λ^*
$\mathbf{s}^k$
μ^k
2
4
3
1
$\mathbf{s}^{k-1}$
$\varsigma_k \mathbf{s}^{k-1}$
λ_k
λ_{k-1}

$\varsigma_k < 0$ $\varsigma_k > 0$

Figure 6.1: Geometric interpretation - acute angle

Lagrange multiplier can be updated through:

$$\mathbf{s}^k = \mu^k + \zeta_k \mathbf{s}^{k-1} \tag{6.1}$$

where the weighting factor ζ_k could be a negative number. In the search direction adjustment scheme discussed in formula (5.9), whether the angle $\angle 3$ is obtuse or not can be determined qualitatively by checking the validity of inequality $\mathbf{s}_{k-1} \cdot \mu_k \leq 0$; however, in the cases shown in Figure 6.1, the selection of weighting factor ζ_k may require a quantitative comparison of those angles, i.e., an estimation of those angles may be required at some accuracy level. It seems to be a necessity to have λ^* to measure (or calculate) those angles, but it is unrealistic before we reach the optimum. This is a key challenge in determining an appropriate weighting factor ζ_k.

Before a breakthrough in making comparison between those angles (without resorting to λ^*), the formula (6.1) may be used when a good estimation of λ^* is available. For example, when λ_k is close to λ^* (e.g., when there is small improvement in Lagrangian function values), the best Lagrange multiplier so far can be used as a "good" estimate of λ^* for evaluating the angles in Figure 6.1.

Bibliography

Andrei, Neculai (2004). On the Complexity of MINOS Package for Linear Programming. *Studies in Informatics and Control* **13**, 35–46.

Baker, Barrie M. and Janice Sheasby (1999). Accelerating the Convergence of Subgradient Optimisation. *European Journal of Operational Research* **117**, 136–144.

Bansal, V., J. D. Perkins, E. N. Pistikopoulos, R. Ross and J. M. G. van Schijndel (2000). Simultaneous Design and Control Optimization under Uncertainty. *Computers & Chemical Engineering* **24**, 261–266.

Barahona, F. and R. Anbil (2000). The Volume Algorithm: Producing Primal Solutions With A Subgradient Method. *Mathematical Programming* **87**, 385–400.

Barrett, George and Stephane Lafortune (2000). Decentralized Supervisory Control with Communicating Controllers. *IEEE Transactions on Automatic Control* **45**, 1620–1638.

Bazaraa, Mokhtar S. and Hanif D. Sherali (1981). On the Choice of Step Size in Subgradient Optimization. *European Journal of Operational Research* **7**, 380–388.

Bazaraa, Mokhtar S. and Jamie J. Goode (1979). A Survey of Various Tactics for Generating Lagrangian Multipliers in the Context of Lagrangian Duality. *European Journal of Operational Research* **3**, 322–338.

Beasley, John E. (1993). *Lagrangian Relaxation (in Modern Heuristic Techniques for Combinatorial Problems, edited by Colin R. Reeves)*. Blackwell Scient. Pub.

Bemporad, A. and M. Morari (1999). Control of Systems Integrating Logic, Dynamics, and Constraints. *Automatica* **35**, 407–427.

Benders, J. F. (1962). Partitioning Procedures for Solving Mixed Variables Programming. *Numerische Mathematik* **4**, 238–252.

Bertsekas, Dimitri P. (1995). *Nonlinear Programming*. 1 ed.. Athena Scientific, Belmont, Massachusetts.

Biegler, Lorenz T. and Ignacio E. Grossmann (2004). Retrospective on Optimization. *Computers & Chemical Engineering* **28**, 1169–1192.

Cai, Ximing, Daene McKinney, Leon Lasdon and David Watkins (2001). Solving Large Nonconvex Water Resources Management Problems Using Generalized Benders Decomposition. *Operations Research* **49:2**, 235–246.

Calamai, Paul H. and Luis N. Vicente (1997). Generating Quadratic Bi-level Programming Test Problems. *ACM Transactions on Mathematical Software* **20**, 103–119.

Camerini, P. M., L. Fratta and F. Maffioli (1975). On Improving Relaxation Methods by Modified Gradient Techniques. *Mathematical Programming Study* **3**, 26–34.

Camponogara, Eduardo, Dong Jia, Bruce H. Krogh and Sarosh Talukdar (2002). Distributed Model Predictive Control. *IEEE Control Systems Magazine* **0272-1708/02**, 44–52.

Castro, J. J. and F. J. Doyle III (2004*a*). A Pulp Mill Benchmark Problem for Control: Application of Plantwide Control Design. *Journal of Process Control* **3**, 329–347.

Castro, J. J. and F. J. Doyle III (2004*b*). A Pulp Mill Benchmark Problem for Control: Problem Description. *Journal of Process Control* **14**, 17–29.

Cheng, R., J. F. Forbes and W. S. Yip (2006*a*). Plant-wide MPC Coordination via Dantzig-Wolfe Decomposition. *Computers & Chemical Engineering, submitted*.

Cheng, R., J. F. Forbes and W. S. Yip (2006*b*). Price-driven Coordination Method for Solving Plant-wide MPC Problems. *Journal of Process Control* **Special Issue**, On line.

Cheng, R., J. F. Forbes, W. S. Yip and J. V. Kresta (2005*a*). Plant-wide MPC: A Cooperative Decentralized Approach. In: *2005 IEEE-IAS Advanced Process Control Applications for Industry Workshop*. May 9-11, 2005, Vancouver, Canada.

Cheng, Ruoyu, J. Fraser Forbes and W. San Yip (2004). Dantzig-Wolfe Decomposition and Large-scale Constrained MPC Problems. In: *7th IFAC Symposium on Dynamics and Control of Process Systems, paper 117*. July 5-7, 2004, Boston, USA.

Cheng, Ruoyu, J. Fraser Forbes and W. San Yip (2005*b*). Price-driven Coordination for Solving Plant-wide MPC Problems. In: *16th IFAC World Congress*. July 4-8, 2005, Prague, Czech.

Chvatal, Vasek (1983). *Linear Programming*. W. H. Freeman and Company, NewYork.

Dantzig, G. B. (1963). *Linear Programming and Extensions*. Princeton University Press.

Dantzig, G. B. and P. Wolfe (1960). Decomposition Principle for Linear Programs. *Operations Research* **8**, 101–111.

Dantzig, George B. and Mukund N. Thapa (2002). *Linear Programming 2: Theory and Extensions*. Springer Verlag.

Davies, Philip I. and Nicholas J. Higham (2000). Numerically Stable Generation of Correlation Matrices and Their Factors. *BIT* **40**, 640–651.

Desaulniers, Guy, Jacques Desrosiers and Marius M. Solomon (2001). Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems. *Essays and Surveys in Metaheuristics* **Chapter 14**, 309–324.

Edgar, Thomas A. and David M. Himmelblau (2001). *Optimization of Chemical Processes*. 2nd ed.. McGraw-Hill Science.

Elbrond, J. and F. Soumis (1987). Towards Integrated Production Planning and Truck Dispatching in Open Pit Mine. *International Journal of Surface Mining, Reclamation and Environment* **1**, 1–6.

Faaland, Bruce H. and Frederick S. Hillier (1979). Interior Path Methods for Heuristic Integer Programming Procedures. *Operations Research* **27**, 1069–1087.

Floudas, Christodoulos A. (1995). *Nonlinear and Mixed-Integer Optimization - Fundamentals and Applications*. 1 ed.. Oxford University Press.

Fumero, Francesca (2001). A Modified Subgradient Optimization Algorithm for Lagrangian Relaxation. *Computers & Operations Research* **28**, 33–52.

Gallestey, E., A. Stothert, D. Gastagnoli, G. Ferrari-Trecate and M. Morari (2003). Using Model Predictive Control and Hybrid Systems for Optimal Scheduling of Industrial Processes. *Automatisierungs technik* **6**, 285–293.

GAMS (2004). CPLEX 9 Manual. *http://www.gams.com/solvers/cplex.pdf* pp. 1–33.

Geoffrion, A. M. (1974). Lagrangean Relaxation for Integer Programming. *Mathematical Programming Study* **2**, 82–114.

Gilmore, P. C. and R. E. Gomory (1961). A Linear Programming Approach to the Cutting Stock Problem. *Operations Research* **9**, 849–859.

Gothe-Lundgren, Maud, J. T. Lundgren and Jan A. Persson (2002). An Optimization Model for Refinery Production Planning. *International Journal of Production Economics* **78**, 255–270.

Goux, Jean-Pierre and Sven Leyffer (2001). Solving Large MINLPs on Computational Grids. *Numerical Analysis Report of Department of Mathematics, University of Dundee* **NA/200**, 1–22.

Grossmann, Ignacio E. and Lorenz T. Biegler (2004). Part II: Future Perspective on Optimization. *Computers & Chemical Engineering* **28**, 1193–1281.

Guta, Berhanu (2003). *Subgradient Optimization Methods in Integer Programming with an Application to a Radiation Therapy Problem*. University of Kaiserslautern.

Hane, Christopher A., Cynthia Barnhart, Ellis L. Johnson, Roy E. Marsten, George L. Nemhauser and Gabrele Sigismondi (1995). The Fleet Assignment Problem: Solving a Large-Scale Integer Program. *Mathematical Programming* **70**, 211–232.

Havlena, V. and J. Lu (2005). A Distributed Automation Framework for Plant-wide Control, Optimization, Scheduling and Planning. In: *16th IFAC World Congress*. July 4-8 2005, Prague, Czech.

Held, M., P. Wolfe and H. P. Crowder (1974). Validation of Subgradient Optimization. *Mathematical Programming* **6**, 62–88.

Hillier, Frederick S. (1969). Efficient Heuristic Procedures for Integer Programming with an Interior. *Operations Research* **17**, 600–637.

Illes, Tibor and Tamas Terlaky (2002). Pivot versus Interior Point Method: Pros and Cons. *European Journal of Operational Research* **140**, 170–190.

Isaksson, A., B. J. Cott, K. Klatt, J. A. Mandler and P. Daoutidis (2005). Panel Discussion: Industrial Perspectives on Process Control. In: *16th IFAC World Congress*. July 4-8, 2005, Prague, Czech.

Jamshidi, Mohammad (1983). *Large-Scale Systems Modeling and Control*. Elsevier Science Publishing Co.,Inc.

Jennergren, L. P. (1973). A Price Schedules Decomposition Algorithm for Linear Programming Problems. *Econometrica* **41**, 965–980.

Jose, Rinaldo A. and Lyle H. Ungar (1998*a*). Auction-driven Coordination for Plantwide Optimization. In: *Foundations of Computer-aided Process Operation (FOCAPO)*. July 5-10, 1998, Snowbird, Utah.

Jose, Rinaldo A. and Lyle H. Ungar (1998*b*). Pricing Interprocess Streams Using Slack Auctions. *AIChE Journal* **46**, 575–587.

Kadam, J. V., M. Schlegel, W. Marguardt, R. L. Tousain, D. H. V. Hesssem, J. V. D. Berg and O. H. Bosgra (2002). A Two-level Strategy of Integrated Dynamic Optimization and Control of Industrial Processes - a Case Study. *European Symposium on Computer Aided Process Engineering, Elsevier* **12**, 511–516.

Kanzow, Christian and Heiko Pieter (1999). Jacobian Smoothing Methods for Nonlinear Complementarity Problems. *SIAM Journal on Optimization* **9**, 342–373.

Kassmann, Dean E., Thomas A. Badgwell and Robert B. Hawkins (2000). Robust Steady-State Target Calculation for Model Predictive Control. *AIChE Journal* **46**, 1007–1024.

Kim, S., H. Ahn and S. C. Cho (1991). Variable Target Value Subgradient Method. *Mathematical Programming* **49**, 359–369.

Kornai, J. and T. Liptak (1965). Two Level Planning. *Econometrica* **33**, 141–168.

Kronsjo, Lydia (1987). *Algorithms: Their Complexity and Efficiency*. 2 ed.. John Wiley & Sons.

Kumar, A. and P. Daoutidis (2002). Nonlinear Dynamics and Control of Process Systems with Recycles. *Journal of Process Control* **12**, 474–484.

Lababidi, Haitham M.S., Samir Kotob and Bader Yousuf (2002). Refinery Advance Process Control Planning System. *Computers & Chemical Engineering* **26**, 1303–1319.

Larsson, T., M. Patriksson and A. B. Stromberg (1996). Conditional Subgradient Optimization: Theory and Applications. *European Journal of Operational Research* **88**, 382–403.

Lasdon, Leon S. (2002). *Optimization Theory for Large Scale Systems*. 2 ed.. Dover Publications INC.

Lestage, Richard, Andre Pomerleau and Daniel Hodouin (2002). Constrained Real-time Optimization of a Grinding Circuit Using Steady-state Linear Programming Supervisory Control. *Power Technology* **124**, 254–263.

Lizotte, Y., E. Bonates and A. Leclerc (1989). Analysis of Truck Dispatching with Dynamic Heuristic Procedure. In: *T. S. Golosinski and V. Srajer (Eds) Off-Highway Haulage in Surface Mines*. Balkema, Rotterdam. pp. 47–55.

Lu, J. Z. (2003). Challenging Control Problems and Emerging Technologies in Enterprise Optimization. *Control Engineering Practice* **11**, 847–858.

McCormick, Garth P. (1983). *Nonlinear Programming – Theory, Algorithms, and Applications*. 1 ed.. John Wiley & Sons, Inc.

Mohideen, M. J., J. D. Perkins and E. N. Pistikopaulos (1997). Towards an Efficient Numerical Procedure for Mixed Integer Optimal Control. *Computers & Chemical Engineering* **21**, 457–462.

Molina, Francisco Walter (1979). A Survey of Resource Directive Decomposition in Mathematical Programming. *Computing Surveys* **11**, 95–104.

Nash, S. G. and A. Sofer (1996). *Linear and Nonlinear Programming*. 1 ed.. McGraw-Hill.

Panne, C. Van De (1975). *Methods for Linear and Quadratic Programming*. 1 ed.. North-Holland Publishing Company-Amsterdam, Oxford.

Pigot, D. (1964). Double Decomposition d'un Programme Linearire. *Actes de la 3e. Conf Int. de Recherche Operationelle* pp. 72–78.

Poljak, B. T. (1969). Minimization of Unsmooth Functionals. *U.S.S.R. Computational Mathematics and Mathematical Physics* **9**, 14–29.

Potra, Florian A. and Stephen J. Wright (2000). Interior-point Methods. *Journal of Computational and Applied Mathematics* **124**, 281–302.

Qin, S. Joe. and Thomas. A. Badgewell (2003). A Survey of Industrial Model Predictive Control Technology. *Control Engineering Practice* **11**, 733–764.

Ralphs, T. K. and M. V. Galati (2003). Decomposition and Dynamic Cut Generation in Integer Programming: Theory and Algorithms. *Industrial and Systems Engineering Technical Report* **03T-005**, 1–32.

Rana, K. (1992). A Decomposition Technique for Mixed Integer Programming Problems. *Computers Operation Research* **19**, 505–519.

Rao, Singiresu S. (1998). *Engineering Optimization - Theory and Practice*. John Wiley & Sons, Inc.

Rosen, J. B. (1964). Primal Partitioning Programming for Block Diagonal Matrices. *Numerische Mathematik* **6**, 250–260.

Sanders, J. L. (1965). A Nonlinear Decomposition Principle. *Operations Research* **13**, 266–268.

Scheiber, Stephen (April 2004). Decentralized Control. *Control Engineering* pp. 44–47.

Sherali, Hanif D. and Gyunghyun Choi (1996). Recovery of Primal Solutions when Using Subgradient Optimization Methods to Solve Lagrangian Duals of Linear Programs. *Operations Research Letters* **19**, 105–113.

Siljak, Dragoslav D. (1991). *Decentralized Control of Complex Systems*. Academic Press.

Stursberg, O. and S. Engell (2002). Optimal Control of Switched Continuous Systems Using Mixed-integer Programming. In: *15th IFAC World Congress, Paper Th-A06-4*. July 21-26, 2002, Barcelona, Spain.

Stursberg, O. and S. Panek (2002). Control of Switched Hybrid Systems Based on Disjunctive Formulations. In: *5th Int. Workshop of Hybrid Systems (HSCC2002)*. Standford (CA), USA.

Ta, C. H., J. V. Kresta, J. F. Forbes and H. J. Marquez (2005). A Stochastic Optimization Approach to Mine Truck Allocation. *International Journal of Surface Mining, Reclamation and Environment* **19**, 162–175.

Ta, Chung Huu (2002). *Optimal Haul Truck Allocation in the Syncrude Mine (Master Thesis)*. University of Alberta.

Temeng, V. A., O. O. Francis and Jr. J. O. Frendewey (1997). Real-time Truck Dispatching Using a Transportation Algorithm. *International Journal of Surface Mining, Reclamation and Environment* **11**, 203–207.

Vanderbei, Robert J. (2001). *Linear Programming: Foundations and Extensions*. 2nd ed.. Kluwer Academic Publishers.

Venkat, Aswin. N., James. B. Rawlings and Stephen J. Wright (2004). Plant-wide Optimal Control with Decentralized MPC. In: *7th IFAC Symposium on Dynamics and Control of Process Systems, paper 190*. July 5-7, 2004, Boston, USA.

Wang, Shih-Ho (2003). An Improved Stepsize of the Subgradient Algorithm for Solving the Lagrangian Relaxation Problem. *Computers and Electrical Engineering* **29**, 245–249.

Whinston, Andrew (1966). A Decomposition Algorithm for Quadratic Programming. *Cowles Foundation Paper* **8**, 112–131.

Wismer, David A. (Editor) (1971). *Optimization Methods for Large-Scale Systems – with Applications*. McGraw-Hill.

Wolbert, D., X. Joulia, B. Koehret and L. T. Biegler (1994). Flowsheet Optimization and Optimal Sensitivity Analysis Using Analytical Derivatives. *Computers & Chemical Engineering* **18**, 1083–1095.

Wolsey, L. A. (1998). *Integer Programming*. 1 ed.. A Wiley-Interscience Publication.

Xi, Y. and T. M. Yegulalp (1993). Optimum Dispatching Algorithm for Anshan Open-pit Mine. In: *24th APCOM Symposium on Application of Computers and Operations Research in the Mineral Industries*. October 31 - November 3, 1993, Montreal, Canada. pp. 426–433.

Ye, Yinyu (2006). Recent Applications of Linear Programming - in Memory of George Dantzig. In: *19th International Symposium on Mathematical Programming (ISMP)*. July 30 - August 4, 2006, Praia Vermelha, Brazil.

Ying, Chao-Ming and Babu Joseph (1999). Performance and Stability Analysis of LP-MPC and QP-MPC Cascade Control Systems. *AIChE Journal* **45**, 1521–1534.

Zhao, X., P. B. Luh and J. Wang (1999). Surrogate Gradient Algorithm for Lagrangian Relaxation. *Journal of Optimization Theory and Applications* **100**, 669–712.

A

Structural Empirical Studies

A.1 Dantzig-Wolfe Decomposition Algorithm

A.1.1 Linear Programming Problem Instance Generation

To generate a test problem instance set, we follow the block-wise structure in (2.26) and (2.27). To generate one LP instance, we take the following steps, assuming the optimization problem is formulated with some scaling operations:

1. Generate p sets of subproblem constraints $\mathbf{B}_i\mathbf{x}_i \leq \mathbf{b}_i$: generate a random vector $\mathbf{x}_i$ with n_i elements in $[1, 10]$ → generate a random $m_i \times n_i$ matrix $\mathbf{B}_i$ with elements in $[10^{-6}, 10^3]$ → calculate $\mathbf{b}_i^o = \mathbf{B}_i\mathbf{x}_i$ → perturb $\mathbf{b}_i = \mathbf{b}_i^o + \alpha\mathbf{b}_i^o$, where α is a m_i vector with randomly generated elements in $[0, 0.5]$, then we have generated subproblem constraints which have feasible solutions.

2. Generate m_0 linking constraints: combine $X = [\mathbf{x}_1, ..., \mathbf{x}_p]$ of a dimension $N \rightarrow$ generate a random $m_0 \times N$ matrix $\mathbf{A}$ with elements in $[10^{-6}, 10^3] \rightarrow$ calculate $\mathbf{b}_0^o = \mathbf{A}X \rightarrow$ perturb $\mathbf{b}_0 = \mathbf{b}_0^o + \beta \mathbf{b}_0^o$, where β is a m_0 vector with randomly generated elements in $[0, 0.5]$.
3. Generate a N vector c with random elements in [0, 10] (in theory, we can generate an unrestricted c vector).

The generated LP instance should have feasible solutions. Degeneracy and cycling is avoided by careful design of the problem instance generation algorithm.

A.1.2 Monte Carlo Simulations

Numerical experiments were designed for the following scenarios:

1. An appropriate reference problem model must be specified. The reference problem size and structure should be a good reference for the comparison experiments, i.e., we can observe the algorithm performance changes when we change the problem with respect to the reference model. In the preliminary study, the reference model is chosen from:

$$\begin{aligned} p &= 17 \\ m_0 &= \lfloor 30R^2/10 \rfloor \\ m_i &= \lfloor 40R^2/10 \rfloor \\ n_i &= \lfloor 30R^2/10 \rfloor \\ R &= \{1, 2, 3, 4, 5, 6, 7\} \end{aligned}$$

   In this case, the overall problem size can be respresented as the number of elements in the coefficient matrix $I = (m_0 + \sum_i^p m_i) \times N$, or in standard LP form $I = (m_0 + \sum_{i=1}^p m_i) \times (\sum_{i=1}^p m_i + m_0 + N)$.

2. For fixed $p = 17$, $|I_i| = 40 \times 30$, we change m_0 in the following way:

$$m_0 = \lfloor 30 \times 2^{R-3} \rfloor, \quad R = \{1, 2, 3, 4, 5, 6, 7\}$$

3. For fixed $p = 17$ and $m_0 = 30$, change subproblem size ($m_i \times n_i$) by factors of 2 to the reference problem model, by changing m_i and n_i:

$$m_i = \lfloor 40 \times 2^{R-3} \rfloor$$
$$n_i = \lfloor 30 \times 2^{R-3} \rfloor, \quad R = \{1, 2, 3, 4, 5, 6, 7\}$$

4. We keep $m_0 = 30$, $m_i = 40$ and $n_i = 30$ constant and change the number of subproblems p:

$$p = 8R + 1, \quad R = \{1, 2, ..., 15\}$$

In this case, we assume a rather well-balanced subproblem load, i.e., $m_1, m_2, ..., m_p$ is in similar order of magnitude and the same to $n_1, n_2, ..., n_p$.

5. By fixing $m_0 = 30$, $\sum_i^p m_i$ and $\sum_i^p n_i$, i.e., we fix the overall problem size, we can study the influence of relative subproblem ratio (RSR). In this case, we change p by combining subproblems into groups following the patterns below:

$$\{1, 1, ..., 1, 1\}, \{2, 2, .., 2, 1\}, \{4, 4, 4, 4, 1\}, \{8, 8, 1\}, \{16, 1\}$$

and

$$\{1, 1, ..., 1, 1\}, \{2, 1, .., 1, 1\}, \{4, 1, ..., 1\}, \{8, 1, ..., 1\}, \{16, 1\}$$

in the above cases, RSR changes from 1 to 16.

A.2 Price-driven Coordination Algorithm

A.2.1 Quadratic Programming Problem Instance Generation

To generate a test problem instance set, we follow the block-wise structure presented in (3.24) to (3.25). To generate a QP instance, we take the following steps, assuming the optimization problem is formulated with some scaling operations:

Phase I: unconstrained QP problem

- Step 1: generation of a positive definite symmetric matrix $\mathbf{Q}_i$ using the following "LDL" decomposition formula:

$$\mathbf{Q}_i = \mathbf{L}_i \mathbf{D}_i \mathbf{L}_i^T \qquad i = 1, 2, ..., p$$

where $\mathbf{D}_i$ is an $(n_i \times n_i)$ diagonal matrix with strictly positive elements, $\mathbf{L}_i$ is a unit lower triangular matrix. Note that, the condition number of the resulting matrix $\mathbf{Q}_i$ can be controlled by the relative ratio of the diagonal elements, and thus we can study the effect of the condition number on the computational performance. Then, the generated matrices will be used to form a block-diagonal Hessian matrix $\mathbf{H}$, which is also a positive definite symmetric matrix of dimension ($N \times N$, where $N = \sum_i n_i$).

- Step 2: randomly generate a trial solution $\mathbf{x}_i^0 \in R^{n_i}$, where the elements of $\mathbf{x}_i^0$ is an arbitrary number in $[10^{-3}, 10^3]$.

- Step 3: generate the linear term coefficient $\mathbf{f}_i$ according to the following formula:

$$\mathbf{f}_i = -\mathbf{Q}_i^T \mathbf{x}_i^0 \quad i = 1, 2, ..., p \quad \text{or} \quad \mathbf{f} = -\mathbf{H}^T \mathbf{x}^0$$

Note that, so far an unconstrained QP problem with a solution $\mathbf{x}^0$ has been generated.

- Step 4: a small perturbation is introduced to the linear term coefficient $\mathbf{f} \in R^N$ by:

$$\mathbf{f} = -\mathbf{H}^T \mathbf{x}^0 + \epsilon_1, \quad \text{where} \quad \epsilon_1 \in [0, 1]^N$$

This will lead to a solution $\mathbf{x}^* \neq \mathbf{x}^0$. The reason to perform this step is to avoid a special situation, where a direct catenation of subproblem solutions is equal to the overall solution, i.e., $[\mathbf{x}_1^0; \mathbf{x}_2^0; ...; \mathbf{x}_p^0] = \mathbf{x}^*$. In fact, in that case, no coordination is required. As can be seen later, the trial solution $\mathbf{x}^0$ will be used to generate constraints, including the linking constraints, thus the resultant optimal solution $\mathbf{x}^*$ will not equal $\mathbf{x}^0$ due to the changes made to $\mathbf{f}$.

Phase II: constrained QP problem

- Step 5: linking constraints are assumed to be equalities, and generated according to the following formula:

$$\mathbf{b}_0 = \mathbf{A}_0 \mathbf{x}^0 \quad \text{where} \quad \mathbf{A}_0 = [\mathbf{A}_1 \mathbf{A}_2 \ldots \mathbf{A}_p]$$

where $\mathbf{A}_i$ is randomly generated with elements $\mathbf{A}_i(j,k) \in [10^{-2}, 10^2]$. In this way, we can guarantee at least a feasible solution that satisfies the linking constraints.

- Step 6: the generation of subproblem (local) constraints are realized through a two-stage procedure, where both the interior solution case and the boundary solution case are considered.

  Stage 1: generate a set of base equality constraints by:

  $$\mathbf{b}_i^0 = \mathbf{B}_i \mathbf{x}_i^0 \quad i = 1, 2, ..., p$$

  where $\mathbf{B}_i$ is randomly generated with elements $\mathbf{B}_i(j,k) \in [10^{-2}, 10^3]$.

  Stage 2: control the location of the optimal solution in the feasible region, by changing $\mathbf{b}_i^0$ in equation (A.2.1) as:

  $$\mathbf{b}_i = \begin{cases} \mathbf{b}_i^0 + \epsilon_2, & \text{interior solution case;} \\ \mathbf{b}_i^0 - \epsilon_2, & \text{boundary solution case.} \end{cases}$$

  where $\epsilon_2 > 0$ contains elements whose values are related to the significance of perturbation in Step 4. By doing this, the resulting local inequality constraints $\mathbf{B}_i \mathbf{x}_i \leq \mathbf{b}_i$ can present different cases when different modification strategies in (A.2.1) are taken. It should be noted that the cases of interior solution and boundary solution are discussed in a reduced space excluding the linking equality constraints.

So far, the generated QP instance has a form of the optimization problem in equations from (3.24) to (3.25), and should have feasible solutions[1].

A.2.2 Monte Carlo Simulations

According to different situations in practice, we design experiments for the following scenarios.

---

[1] According to our experience, more than 95% of the generated QP problems have optimal solutions, and this satisfies the requirements of the simulation studies. For more advanced instance generation techniques, interested researchers may refere to (Calamai and Vicente, 1997; Davies and Higham, 2000).

1. We need to start from an appropriate reference problem model. The reference problem size and structure should be a good reference for the comparison experiments, i.e., we can observe the algorithm performance changes when we change the problem with respect to the reference model. In the preliminary study, the reference model is chosen from:

$$\begin{aligned} p &= 17 \\ m_0 &= \lfloor 10R^2/10 \rfloor \\ m_i &= \lfloor 15R^2/10 \rfloor \\ n_i &= \lfloor 10R^2/10 \rfloor \\ R &= \{1,2,3,4,5,6,7\} \end{aligned}$$

   In this case, the overall problem size can be respresented as the number of elements in the coefficient matrix $I = (m_0 + \sum_i^p m_i) \times N$.

2. For fixed $p = 17$, $|I_i| = 30 \times 20$, we change m_0 in the following way:

$$m_0 = \lfloor 10 \times 2^{R-3} \rfloor, \quad R = \{1,2,3,4,5,6,7\}$$

3. For fixed $p = 17$ and $m_0 = 20$, change subproblem size ($m_i \times n_i$) by factors of 2 to the reference problem model, by changing m_i and n_i:

$$\begin{aligned} m_i &= \lfloor 15 \times 2^{R-3} \rfloor \\ n_i &= \lfloor 10 \times 2^{R-3} \rfloor, \quad R = \{1,2,3,4,5\} \end{aligned}$$

4. We keep $m_0 = 20$, $m_i = 30$ and $n_i = 20$ constant and change the number of subproblems p:

$$\begin{aligned} p &= 8R + 1, \quad R = \{1,2,...,11\} \text{ for interior cases} \\ p &= 8R + 1, \quad R = \{1,2,...,9\} \text{ for boundary cases} \end{aligned}$$

   where we assume a rather well-balanced subproblem load, i.e., $m_1, m_2, ..., m_p$ is in similar order of magnitude and the same to $n_1, n_2, ..., n_p$.

5. If we fix $m_0 = 20$, $\sum_i^p m_i$ and $\sum_i^p n_i$, i.e., we fix the overall problem size, we can study the influence of relative subproblem ratio (RSR). In this case, we change p by combining subproblems into groups following the patterns below:

$$\{1, 1, ..., 1, 1\}, \{2, 2, .., 2, 1\}, \{4, 4, 4, 4, 1\}, \{8, 8, 1\}, \{16, 1\}$$

   in the above cases, RSR is changing from 1 to 16.

A.3 Improved Subgradient Optimization Algorithm

A.3.1 Binary Integer Programming Problem Instance Generation

To generate a test problem instance set, we follow the block-wise structure presented in (5.32). To generate a BIP instance, we take the following steps, assuming the optimization problem is formulated with some scaling operations:

1. For a given input parameter n_i, generate a set of random vectors $\mathbf{x}_i$ for $i = 1, ..., p$ with binary elements $\in \{0, 1\}$.

2. Generation of p sets of subproblem constraints $\mathbf{B}_i\mathbf{x}_i \leq \mathbf{b}_i$: generate a random $m_i \times n_i$ matrix $\mathbf{B}_i$ with elements in $[10^{-3}, 10^2]$ → calculate $\mathbf{b}_i^o = \mathbf{B}_i\mathbf{x}_i$ → perturb $\mathbf{b}_i = \mathbf{b}_i^o + \alpha\mathbf{b}_i^o$, where α is a m_i vector with randomly generated elements in $[0, 0.5]$, then we have generated subproblem constraints which have feasible solutions.

3. Generation of m_0 linking constraints: combine $X = [\mathbf{x}_1, ..., \mathbf{x}_p]$ of a dimension N → generate a random $m_0 \times N$ matrix $\mathbf{A}$ with elements in $[10^{-3}, 10^2]$ → calculate $\mathbf{b}_0^o = \mathbf{A}X$ → perturb $\mathbf{b}_0 = \mathbf{b}_0^o + \beta\mathbf{b}_0^o$, where β is a m_0 vector with randomly generated elements in $[0, 0.5]$.

4. Generate a N vector $\mathbf{c}$ with random elements in $[10^{-3}, 10]$ (in theory, we can generate an unrestricted $\mathbf{c}$ vector).

5. Reformulation of linking constraints (scaling of RHS of the linking constraints): since it is quite common to normalize the RHS of the linking constraints in the

applications of subgradient optimization (Beasley, 1993), the coefficients of matrix A is divided by the corresponding elements of vector $\mathbf{b}_0$, and therefore the RHS of linking constraints is normalized to 1.

A.3.2 Monte Carlo Simulations

Following the same structural analysis strategy, numerical experiments were designed for the following scenarios:

1. An appropriate reference problem model must be specified. With a preliminary empirical study, the reference problem size is chosen from:

$$\begin{aligned} p &= 17 \\ m_0 &= \lfloor 2 \times 2^{R-2} \rfloor \\ m_i &= \lfloor 3 \times 2^{R-2} \rfloor \\ n_i &= \lfloor 2 \times 2^{R-2} \rfloor \\ R &= \{2, 3, 4, 5, 6, 7\} \end{aligned}$$

   In this case, the overall problem size can be respresented as the number of elements in the coefficient matrix $I = (m_0 + \sum_i^p m_i) \times N$.

2. Scenario 1: for fixed $p = 17$, $|I_i| = 12 \times 8$, we change m_0 in the following way:

$$m_0 = 10 + 20 \times (R - 1), \quad R = \{1, 2, ..., 10\}$$

3. Scenario 2: for fixed $p = 17$ and $m_0 = 10$, change subproblem size $(m_i \times n_i)$ through changing the reference model parameter R:

$$\begin{aligned} m_i &= \lfloor 3 \times 2^{R-2} \rfloor \\ n_i &= \lfloor 2 \times 2^{R-2} \rfloor, \quad R = \{1, 2, ..., 7\} \end{aligned}$$

4. Scenario 3: we keep $m_0 = 10$, $m_i = 12$ and $n_i = 8$ constant and change the number of subproblems p:

$$p = R^2, \quad R = \{1, 2, ..., 9\}$$

In this case, we assume a rather well-balanced subproblem load, i.e., $m_1, m_2, ..., m_p$ is in similar order of magnitude and the same to $n_1, n_2, ..., n_p$.

5. Scenario 4: by fixing $m_0 = 10$, $\sum_i^p m_i$ and $\sum_i^p n_i$, i.e., we fix the overall problem size, we can study the influence of relative subproblem ratio (RSR). In this case, we change p by combining subproblems into groups following the patterns below:

$$\{1, 1, ..., 1, 1\}, \{2, 2, .., 2, 1\}, \{4, 4, 4, 4, 1\}, \{8, 8, 1\}, \{16, 1\}$$

in the above cases, RSR changes from 1 to 16.